World Population Analysis

In [65]:
```
import numpy as np
import pandas as pd
import seaborn as sns
from plotly.subplots import make_subplots
import matplotlib.pyplot as plt
from itables import init_notebook_mode
from itables import show
import plotly.express as px
from plotly.offline import iplot, init_notebook_mode

from plotly.offline import download_plotlyjs,init_notebook_mode, pl
init_notebook_mode(connected=True)
cf.go_offline()

import warnings
warnings.filterwarnings('ignore')
```

Importing Dataset

In [45]:
```
#loading dataset in pandas
data = pd.read_csv('world_population.csv')
```

In [46]:
```
#check first five rows
data.head()
```

Out[46]:

	Rank	CCA3	Country/Territory	Capital	Continent	2022 Population	2020 Population	2015 Population	I
0	36	AFG	Afghanistan	Kabul	Asia	41128771	38972230	33753499	
1	138	ALB	Albania	Tirana	Europe	2842321	2866849	2882481	
2	34	DZA	Algeria	Algiers	Africa	44903225	43451666	39543154	
3	213	ASM	American Samoa	Pago Pago	Oceania	44273	46189	51368	
4	203	AND	Andorra	Andorra la Vella	Europe	79824	77700	71746	

In [47]:
```
#check last five rows
data.tail()
```

Out[47]:

	Rank	CCA3	Country/Territory	Capital	Continent	2022 Population	2020 Population	2015 Population
229	226	WLF	Wallis and Futuna	Mata-Utu	Oceania	11572	11655	12182
230	172	ESH	Western Sahara	El Aaiún	Africa	575986	556048	491824
231	46	YEM	Yemen	Sanaa	Asia	33696614	32284046	28516545
232	63	ZMB	Zambia	Lusaka	Africa	20017675	18927715	16248230
233	74	ZWE	Zimbabwe	Harare	Africa	16320537	15669666	14154937

In [48]:
```
#check shape
data.shape
```

Out[48]: (234, 17)

In [49]:
```
#check more info
data.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 234 entries, 0 to 233
Data columns (total 17 columns):
 #   Column                      Non-Null Count  Dtype  
---  ------                      --------------  -----  
 0   Rank                        234 non-null    int64  
 1   CCA3                        234 non-null    object 
 2   Country/Territory           234 non-null    object 
 3   Capital                     234 non-null    object 
 4   Continent                   234 non-null    object 
 5   2022 Population             234 non-null    int64  
 6   2020 Population             234 non-null    int64  
 7   2015 Population             234 non-null    int64  
 8   2010 Population             234 non-null    int64  
 9   2000 Population             234 non-null    int64  
 10  1990 Population             234 non-null    int64  
 11  1980 Population             234 non-null    int64  
 12  1970 Population             234 non-null    int64  
 13  Area (km²)                  234 non-null    int64  
 14  Density (per km²)           234 non-null    float64
 15  Growth Rate                 234 non-null    float64
 16  World Population Percentage 234 non-null    float64
dtypes: float64(3), int64(10), object(4)
memory usage: 31.2+ KB
```

In [50]:
```
#mathmatic realtion
data.describe()
```

Out[50]:

	Rank	2022 Population	2020 Population	2015 Population	2010 Population	2000 Population
count	234.000000	2.340000e+02	2.340000e+02	2.340000e+02	2.340000e+02	2.340000e+02
mean	117.500000	3.407441e+07	3.350107e+07	3.172996e+07	2.984524e+07	2.626947e+07
std	67.694165	1.367664e+08	1.355899e+08	1.304050e+08	1.242185e+08	1.116982e+08
min	1.000000	5.100000e+02	5.200000e+02	5.640000e+02	5.960000e+02	6.510000e+02
25%	59.250000	4.197385e+05	4.152845e+05	4.046760e+05	3.931490e+05	3.272420e+05
50%	117.500000	5.559944e+06	5.493074e+06	5.307400e+06	4.942770e+06	4.292907e+06
75%	175.750000	2.247650e+07	2.144798e+07	1.973085e+07	1.915957e+07	1.576230e+07
max	234.000000	1.425887e+09	1.424930e+09	1.393715e+09	1.348191e+09	1.264099e+09

In [51]:
```
#check corr realtion
data.corr()
```

Out[51]:

	Rank	2022 Population	2020 Population	2015 Population	2010 Population	2000 Population	19 Populati
Rank	1.000000	-0.358361	-0.355854	-0.351222	-0.347461	-0.341057	-0.3361
2022 Population	-0.358361	1.000000	0.999946	0.999490	0.998629	0.994605	0.9872
2020 Population	-0.355854	0.999946	1.000000	0.999763	0.999105	0.995583	0.9887
2015 Population	-0.351222	0.999490	0.999763	1.000000	0.999783	0.997340	0.9915
2010 Population	-0.347461	0.998629	0.999105	0.999783	1.000000	0.998593	0.9939
2000 Population	-0.341057	0.994605	0.995583	0.997340	0.998593	1.000000	0.9983
1990 Population	-0.336152	0.987228	0.988724	0.991594	0.993929	0.998336	1.0000
1980 Population	-0.335246	0.980285	0.982121	0.985724	0.988786	0.995160	0.9990
1970 Population	-0.335379	0.973162	0.975254	0.979414	0.983042	0.990956	0.9966
Area (km²)	-0.383774	0.453411	0.454993	0.458240	0.461936	0.473933	0.4867
Density (per km²)	0.129436	-0.027618	-0.027358	-0.026857	-0.026505	-0.026139	-0.0262
Growth Rate	-0.224561	-0.020863	-0.025116	-0.032154	-0.037983	-0.050515	-0.0623
World Population Percentage	-0.358464	0.999999	0.999944	0.999487	0.998626	0.994598	0.9872

In [52]:
```
#check missing values
data.isnull().sum()
```

Out[52]:
```
Rank                              0
CCA3                              0
Country/Territory                 0
Capital                           0
Continent                         0
2022 Population                   0
2020 Population                   0
2015 Population                   0
2010 Population                   0
2000 Population                   0
1990 Population                   0
1980 Population                   0
1970 Population                   0
Area (km²)                        0
Density (per km²)                 0
Growth Rate                       0
World Population Percentage       0
dtype: int64
```

Data Processing

In [53]:
```
# renaming 'Country/Territory' to 'Country'

df.rename(columns={'Country/Territory':'Country'}, inplace = True)
```

In [54]:
```
# renaming year columns from "Year Population" to just "Year"

for col in df.columns:
    if 'Population' and '0' in col:
        df = data.rename(columns={col: col.split(' ')[0]})

df.head(3)
```

Out[54]:

	Rank	CCA3	Country/Territory	Capital	Continent	2022 Population	2020 Population	2015 Population	F
0	36	AFG	Afghanistan	Kabul	Asia	41128771	38972230	33753499	
1	138	ALB	Albania	Tirana	Europe	2842321	2866849	2882481	
2	34	DZA	Algeria	Algiers	Africa	44903225	43451666	39543154	

In [5]:
```
# let's check to see how many null objects we have in the dataset

df.isnull().sum()
```

Out[5]:
```
Rank                            0
CCA3                            0
Country                         0
Capital                         0
Continent                       0
2022                            0
2020                            0
2015                            0
2010                            0
2000                            0
1990                            0
1980                            0
1970                            0
Area (km²)                      0
Density (per km²)               0
Growth Rate                     0
World Population Percentage     0
dtype: int64
```

In [6]:
```
# looking for duplicates

df.duplicated().sum()
```

Out[6]:
```
0
```

In [7]:
```
# getting acquinted with our dataset

df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 234 entries, 0 to 233
Data columns (total 17 columns):
 #   Column                          Non-Null Count  Dtype  
---  ------                          --------------  -----  
 0   Rank                            234 non-null    int64  
 1   CCA3                            234 non-null    object 
 2   Country                         234 non-null    object 
 3   Capital                         234 non-null    object 
 4   Continent                       234 non-null    object 
 5   2022                            234 non-null    int64  
 6   2020                            234 non-null    int64  
 7   2015                            234 non-null    int64  
 8   2010                            234 non-null    int64  
 9   2000                            234 non-null    int64  
 10  1990                            234 non-null    int64  
 11  1980                            234 non-null    int64  
 12  1970                            234 non-null    int64  
 13  Area (km²)                      234 non-null    int64  
 14  Density (per km²)               234 non-null    float64
 15  Growth Rate                     234 non-null    float64
 16  World Population Percentage     234 non-null    float64
dtypes: float64(3), int64(10), object(4)
memory usage: 31.2+ KB
```

In [8]:
```
df.nunique()
```

Out[8]:
```
Rank                            234
CCA3                            234
Country                         234
Capital                         234
Continent                         6
2022                            234
2020                            234
2015                            234
2010                            234
2000                            234
1990                            234
1980                            234
1970                            234
Area (km²)                      233
Density (per km²)               234
Growth Rate                     180
World Population Percentage      70
dtype: int64
```

Column names that are integers (such as the years) might introduce some confusion. For example, when we are referencing the year 2010, one might confuse that when the 2010th positional index.

To avoid this ambuigity, let's convert the column names into strings: '1970' to '2010'.

In [9]:
```
# converting the column names into strings

df.columns = list(map(str, df.columns))
```

Since we converted the years to string, let's declare a variable that will allow us to easily call upon the full range of years

In [10]:
```
years = list(map(str, (1970, 1980, 1990, 2000, 2010, 2015, 2020, 20
years
```

Out[10]:
```
['1970', '1980', '1990', '2000', '2010', '2015', '2020', '2022']
```

In [11]:
```
# let's view our statistical summary

df.describe().T.sort_values(ascending=0, by="mean")
```

Out[11]:

	count	mean	std	min	25%	50%	
2022	234.0	3.407441e+07	1.367664e+08	510.0000	419738.500000	5.559944e+06	2.2
2020	234.0	3.350107e+07	1.355899e+08	520.0000	415284.500000	5.493074e+06	2.1
2015	234.0	3.172996e+07	1.304050e+08	564.0000	404676.000000	5.307400e+06	1.9
2010	234.0	2.984524e+07	1.242185e+08	596.0000	393149.000000	4.942770e+06	1.9
2000	234.0	2.626947e+07	1.116982e+08	651.0000	327242.000000	4.292907e+06	1.5
1990	234.0	2.271022e+07	9.783217e+07	700.0000	264115.750000	3.825410e+06	1.1
1980	234.0	1.898462e+07	8.178519e+07	733.0000	229614.250000	3.141146e+06	9.8
1970	234.0	1.578691e+07	6.779509e+07	752.0000	155997.000000	2.604830e+06	8.8
Area (km²)	234.0	5.814494e+05	1.761841e+06	1.0000	2650.000000	8.119950e+04	4.3
Density (per km²)	234.0	4.521270e+02	2.066122e+03	0.0261	38.417875	9.534675e+01	2.3
Rank	234.0	1.175000e+02	6.769417e+01	1.0000	59.250000	1.175000e+02	1.7
Growth Rate	234.0	1.009577e+00	1.338498e-02	0.9120	1.001775	1.007900e+00	1.0
World Population Percentage	234.0	4.270513e-01	1.714977e+00	0.0000	0.010000	7.000000e-02	2.

In [12]:
```
# creating dataframe for 'Continent'

continent_df = df.groupby(by='Continent').sum()
continent_df.head(3)
```

Out[12]:

Continent	Rank	2022	2020	2015	2010	2000	199
Africa	5253	1426730932	1360671810	1201102442	1055228072	818946032	6381506
Asia	3878	4721383274	4663086535	4458250182	4220041327	3735089604	321056357
Europe	6225	743147538	745792196	741535608	735613934	726093423	7203207

In [13]:
```
# creating dataframe for 'Country'

country_df = df.groupby(by='Country').sum()
country_df.head(3)
```

Out[13]:

Country	Rank	2022	2020	2015	2010	2000	1990	198
Afghanistan	36	41128771	38972230	33753499	28189672	19542982	10694796	1248663
Albania	138	2842321	2866849	2882481	2913399	3182021	3295066	294165
Algeria	34	44903225	43451666	39543154	35856344	30774621	25518074	1873937

Exploratory Data Analysis and Visualization

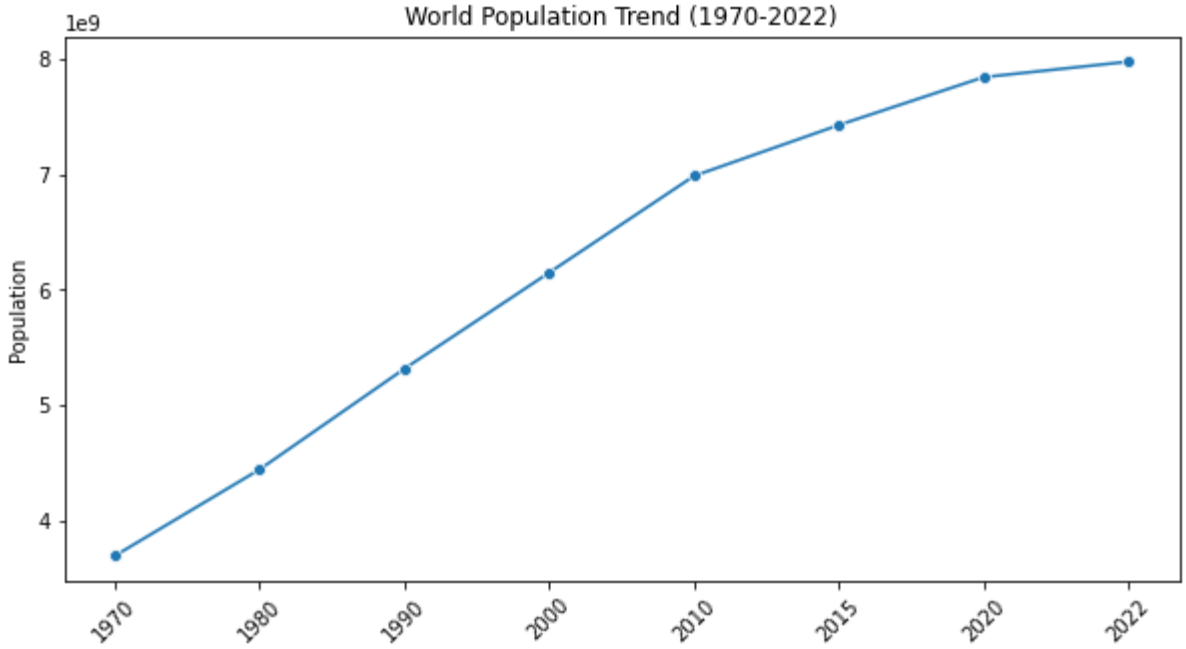
World Population EDA

In [14]:
```
# current world population 2022

df['2022'].sum()
```

Out[14]: 7973413042

Current world population is 7,973,413,042

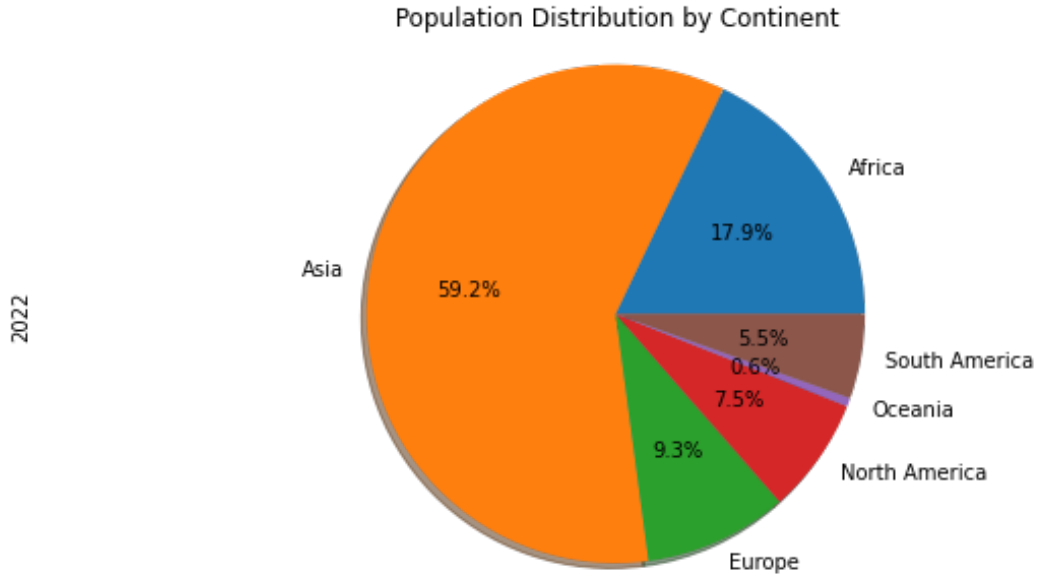
In [15]:
```
# plotting world population trend since 1970

plt.subplots(figsize=(10,5))
trend = df.iloc[:,5:13].sum()[::-1]
sns.lineplot(x=trend.index, y=trend.values, marker="o")
plt.xticks(rotation=45)
plt.ylabel("Population")
plt.title("World Population Trend (1970-2022)")
plt.show()
```



In [16]:

```
#  plotting current world population on map

import pycountry
countries = {}
for country in pycountry.countries:
    countries[country.name] = country.alpha_3


# get average of a list
def Average(list):
    return sum(list) / len(list)

df_rc2022 = df.loc[:,["CCA3", "Country","2022"]]
df_rc2022["CCA3"] = [countries.get(x, 'Unknown code') for x in df_r

fig = px.choropleth(df_rc2022, locations="CCA3",
                    hover_name="Country",
                    hover_data=df_rc2022.columns,
                    color="2022",
                    color_continuous_scale="Viridis",
                    range_color=(min(df_rc2022["2022"]), max(df_rc2
                    projection="natural earth"

                   )

fig.update_layout(margin={"r":5,"t":0,"l":5,"b":0})
fig.show()
```

In [17]:
```
# let's see pie chart distribution for continent_df

continent_df['2022'].plot(kind = 'pie', figsize=(10,5), shadow=True
plt.title(' Population Distribution by Continent')
plt.axis('equal')
plt.show()
```



In [18]:
```
# let's see world population by continent

fig = px.bar(data_frame= df.groupby('Continent' , as_index= False).
fig.update_layout(title= 'Current (2022) World Population per Conti
fig.show()
```

In [19]:
```
# let's see number of countries per continent

df_country=df['Continent'].value_counts()
fig=px.bar(x=df_country.index,
           y=df_country.values,
           color=df_country.index,
           color_discrete_sequence=px.colors.sequential.YlOrRd,
           text=df_country.values,
           title= 'Number of Countries By Continent')

fig.update_layout(xaxis_title="Countries",
                  yaxis_title="Count")

fig.show()
```

Population Growth Rate

In [20]:
```
# average population growth rate

df['Growth Rate'].mean()
```

Out[20]:
```
1.0095773504273504
```

Since 1972 (50 years ago), the world population growth rate declined from around 2% per year to under 1.0% per year.

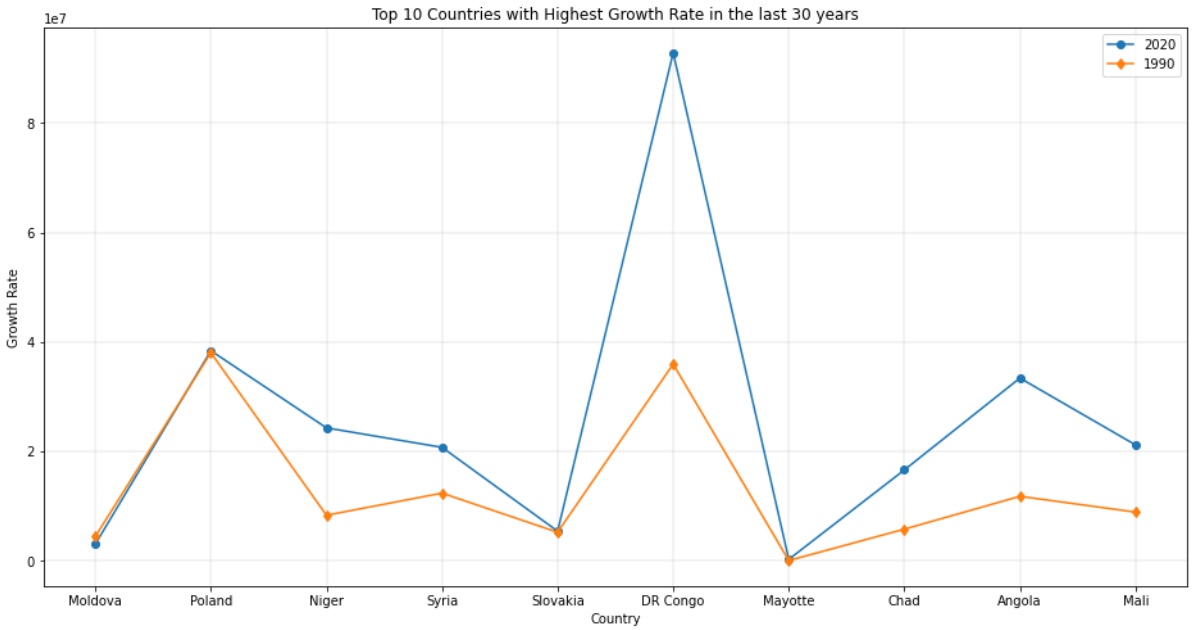
In [21]:
```
# plotting population Growth Rate on map

fig = px.choropleth(df,
                    locations='Country',
                    locationmode='country names',
                    color='Growth Rate',
                    color_continuous_scale='Viridis',
                    template='plotly',
                    title = 'Growth Rate')

fig.update_layout(font = dict(size = 17, family="Gothic"))
```

In [22]:
```
# creating dataframe for top 10 countries with highest growth rate

gwr_top10 = df.sort_values(by='Country').sort_values(by='Growth Rat

gwr_top10.head(3)
```

Out[22]:

	Rank	CCA3	Country	Capital	Continent	2022	2020	2015	2010
133	135	MDA	Moldova	Chisinau	Europe	3272996	3084847	3277388	3678186
164	37	POL	Poland	Warsaw	Europe	39857145	38428366	38553146	38597353
148	54	NER	Niger	Niamey	Africa	26207977	24333639	20128124	16647543

In [23]:
```
# plotting top 10 highest growth rate countries in the last 30 year

fig, ax = plt.subplots(figsize=(16,8))
plt.plot(gwr_top10['Country'], gwr_top10['2020'], label='2020', mar
plt.plot(gwr_top10['Country'], gwr_top10['1990'], label='1990', mar

plt.xlabel('Country')
plt.ylabel('Growth Rate')
plt.grid(linewidth=0.3)
plt.title('Top 10 Countries with Highest Growth Rate in the last 30
plt.legend()
plt.show()
```



Population Decade-By-Decade Percent Change

In [24]:
```
# creating dataframe for population difference decade-by-decade per

pop_diff = df.groupby('Continent')[['1970','1980', '1990', '2000',

pop_diff.head(3)
```

Out[24]:

	Continent	1970	1980	1990	2000	2010	2020
0	Africa	365444348	481536377	638150629	818946032	1055228072	1360671810
1	Asia	2144906290	2635334228	3210563577	3735089604	4220041327	4663086535
2	Europe	655923991	692527159	720320797	726093423	735613934	745792196

In [25]:
```
# finding the population decade-by-decade percent change

pop_diff['70s'] = pop_diff['1970']/pop_diff['1980']*100
pop_diff['80s'] = pop_diff['1980']/pop_diff['1990']*100
pop_diff['90s'] = pop_diff['1990']/pop_diff['2000']*100
pop_diff['00s'] = pop_diff['2000']/pop_diff['2010']*100
pop_diff['10s'] = pop_diff['2010']/pop_diff['2020']*100

pop_diff.head(3)
```

Out[25]:

	Continent	1970	1980	1990	2000	2010	2020
0	Africa	365444348	481536377	638150629	818946032	1055228072	1360671810
1	Asia	2144906290	2635334228	3210563577	3735089604	4220041327	4663086535
2	Europe	655923991	692527159	720320797	726093423	735613934	745792196

In [26]:
```
# creating dataframe for decade-by-decade

decade_diff = pop_diff.groupby('Continent')[['70s','80s', '90s', '0

decade_diff
```

Out[26]:

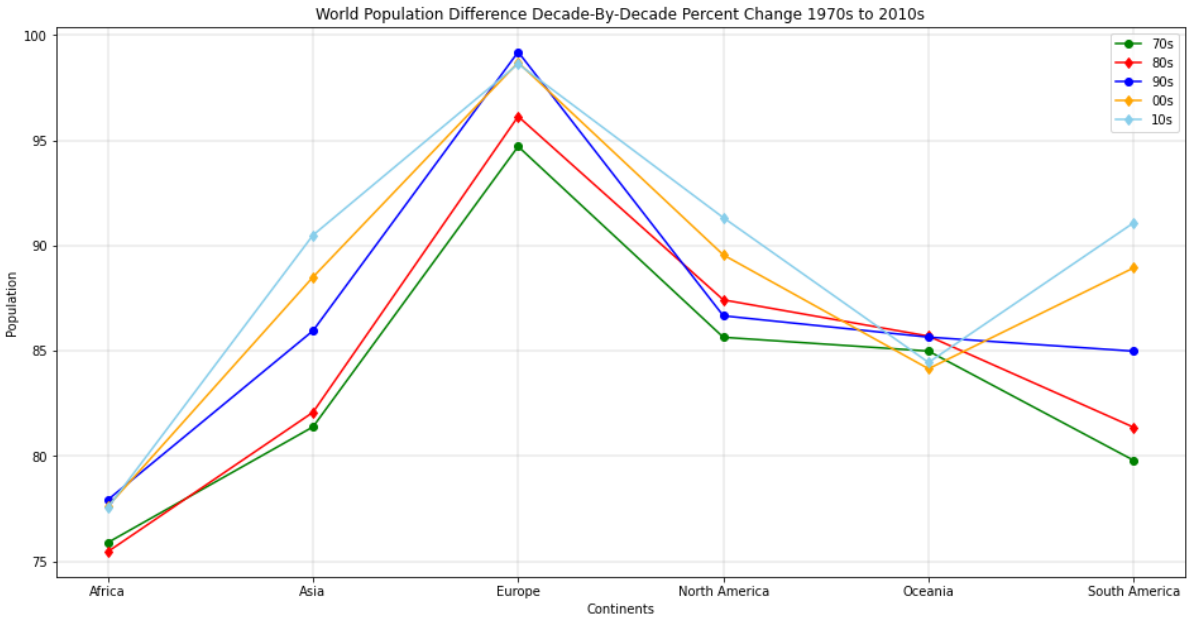
	Continent	70s	80s	90s	00s	10s
0	Africa	75.891327	75.458106	77.923404	77.608439	77.551990
1	Asia	81.390295	82.083228	85.956802	88.508366	90.498885
2	Europe	94.714551	96.141492	99.204975	98.705773	98.635242
3	North America	85.647649	87.425282	86.667926	89.561653	91.330736
4	Oceania	84.991562	85.702934	85.654845	84.152162	84.452244
5	South America	79.799805	81.370326	84.987780	88.947756	91.089429

In [27]:
```
# let's see decade_diff statistical summary quickly

decade_diff.describe()
```

Out[27]:

	70s	80s	90s	00s	10s
count	6.000000	6.000000	6.000000	6.000000	6.000000
mean	83.739198	84.696895	86.732622	87.914025	88.926421
std	6.454366	6.966887	6.898899	6.941975	7.163646
min	75.891327	75.458106	77.923404	77.608439	77.551990
25%	80.197428	81.548551	85.154546	85.241213	85.963904
50%	83.190928	83.893081	85.805823	88.728061	90.794157
75%	85.483627	86.994695	86.490145	89.408178	91.270409
max	94.714551	96.141492	99.204975	98.705773	98.635242

In [28]:
```
# plotting wolrd population difference decade-by-decade percent cha

fig, ax = plt.subplots(figsize=(16,8))
plt.plot(decade_diff['Continent'], decade_diff['70s'], label='70s',
plt.plot(decade_diff['Continent'], decade_diff['80s'], label='80s',
plt.plot(decade_diff['Continent'], decade_diff['90s'], label='90s',
plt.plot(decade_diff['Continent'], decade_diff['00s'], label='00s',
plt.plot(decade_diff['Continent'], decade_diff['10s'], label='10s',
plt.grid(linewidth=0.4)
plt.title("World Population Difference Decade-By-Decade Percent Cha
plt.xlabel('Continents')
plt.ylabel('Population')
plt.legend()
plt.show()
```



Area

In [29]:
```
# plotting total area distribution by continents

import plotly.graph_objects as go

df_cont= df['Continent'].unique()

tot_area_cont = []

for each in df_cont:
    df_area = df[df.Continent == each]
    area = sum(df_area["Area (km²)"])
    tot_area_cont.append(area)

tot_area_cont = pd.DataFrame(tot_area_cont)
df_area = pd.DataFrame(df_cont, columns = ["continent"])
df_area["total"] = tot_area_cont

fig = go.Figure(data=[go.Pie(labels=df_area.continent, values=df_ar
                             insidetextorientation='radial'
                            )])
fig.show()
```

In [30]:
```
# plotting Area distribution on map by country

fig = px.choropleth(df,
                    locations='Country',
                    locationmode='country names',
                    color='Area (km²)',
                    color_continuous_scale='Viridis',
                    template='plotly',
                    title = 'Area (km²)')

fig.update_layout(font = dict(size = 17, family="Gothic"))
```

Top 10 Countries With Most Population

Firstly, let's copy our dataframe 'df' to 'df_copy' so we can place 'Country' as the index to avoid affecting other analysis negatively.

In [31]:
```
# copying dataframe 'df' to 'df_copy'

df_copy = df.copy()
df_copy.head(3)
```

Out[31]:

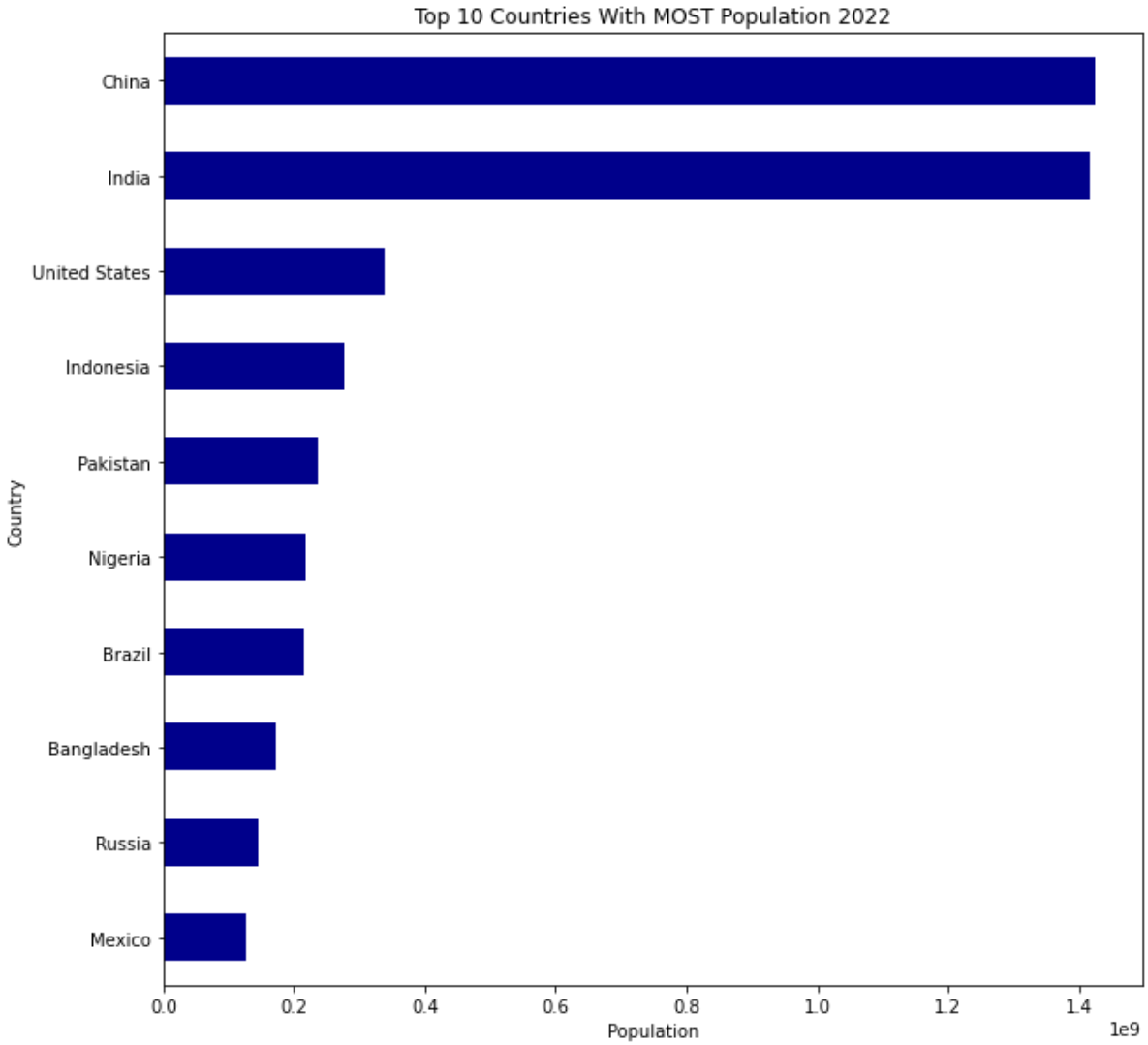
	Rank	CCA3	Country	Capital	Continent	2022	2020	2015	2010
0	36	AFG	Afghanistan	Kabul	Asia	41128771	38972230	33753499	28189672
1	138	ALB	Albania	Tirana	Europe	2842321	2866849	2882481	2913399
2	34	DZA	Algeria	Algiers	Africa	44903225	43451666	39543154	35856344

In [32]:
```
df_copy.set_index('Country', inplace=True)
```

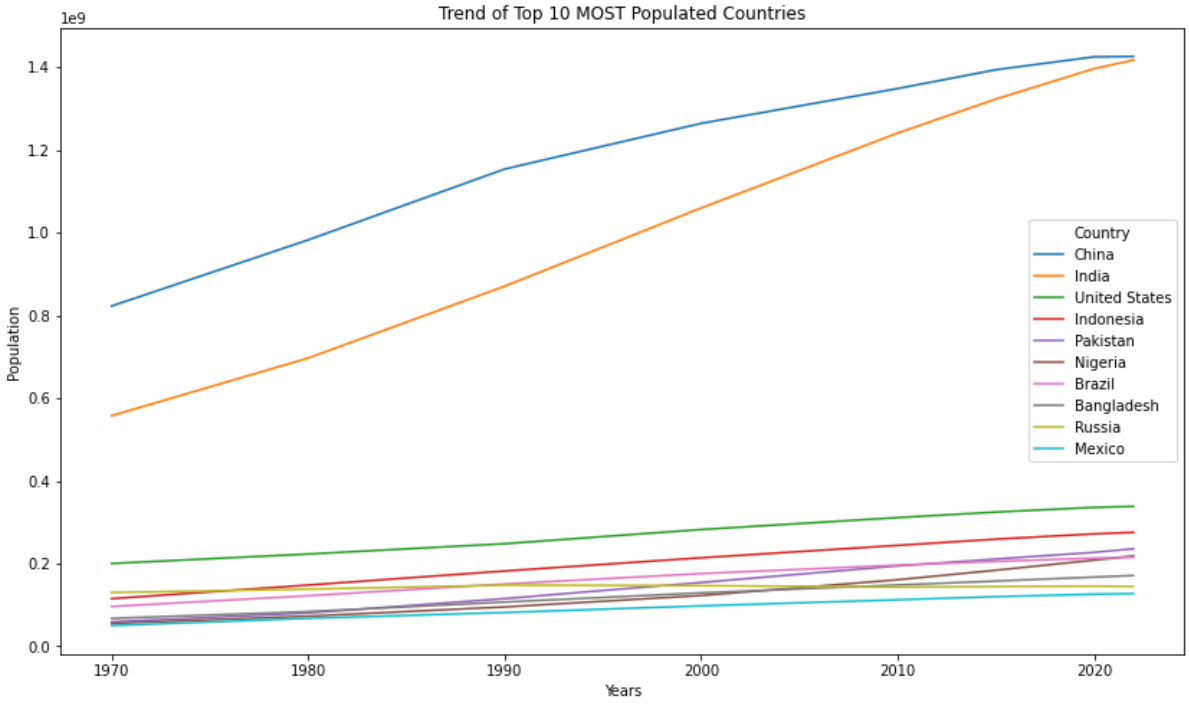
In [33]:
```
df_copy.sort_values(by='2022', ascending=True, inplace=True)

df_top10 = df_copy['2022'].tail(10)
df_top10
```

Out[33]:
```
Country
Mexico            127504125
Russia            144713314
Bangladesh        171186372
Brazil            215313498
Nigeria           218541212
Pakistan          235824862
Indonesia         275501339
United States     338289857
India            1417173173
China            1425887337
Name: 2022, dtype: int64
```

In [34]:
```
# plotting top 10 MOST populated countries

df_top10.plot(kind='barh', figsize=(10, 10), color='darkblue')
plt.xlabel('Population')
plt.title('Top 10 Countries With MOST Population 2022')

plt.show()
```



In [35]:
```
# plotting top 10 population trend

inplace = True
df_copy.sort_values(by='2022', ascending=False, axis=0, inplace=Tru

df_top_10 = df_copy.head(10)

df_top_10 = df_top_10[years].transpose()

df_top_10.index = df_top_10.index.map(int)
df_top_10.plot(kind='line', figsize=(14, 8))

plt.title('Trend of Top 10 MOST Populated Countries')
plt.ylabel('Population')
plt.xlabel('Years')
plt.show()
```

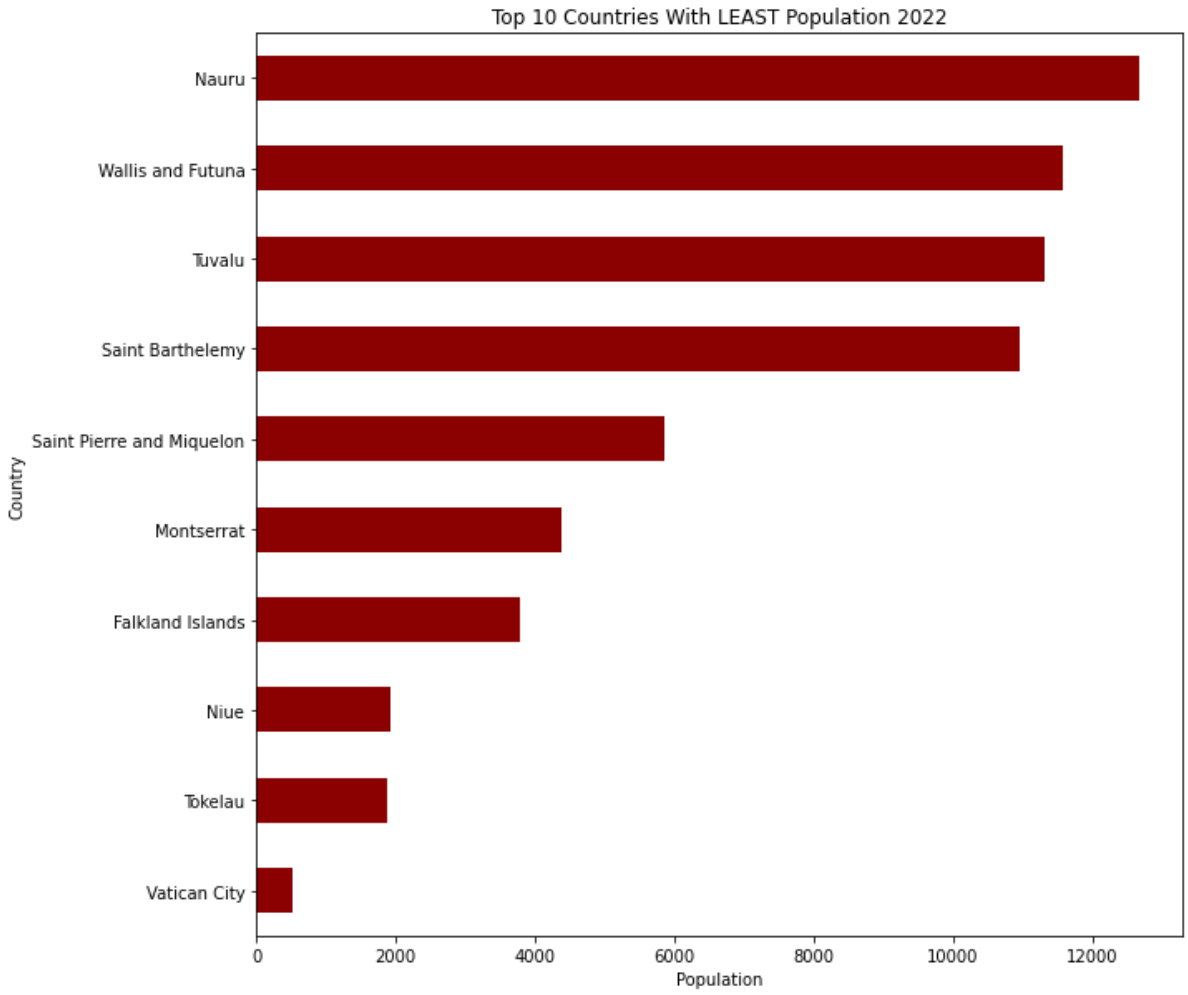


Top 10 Countries With Least Population

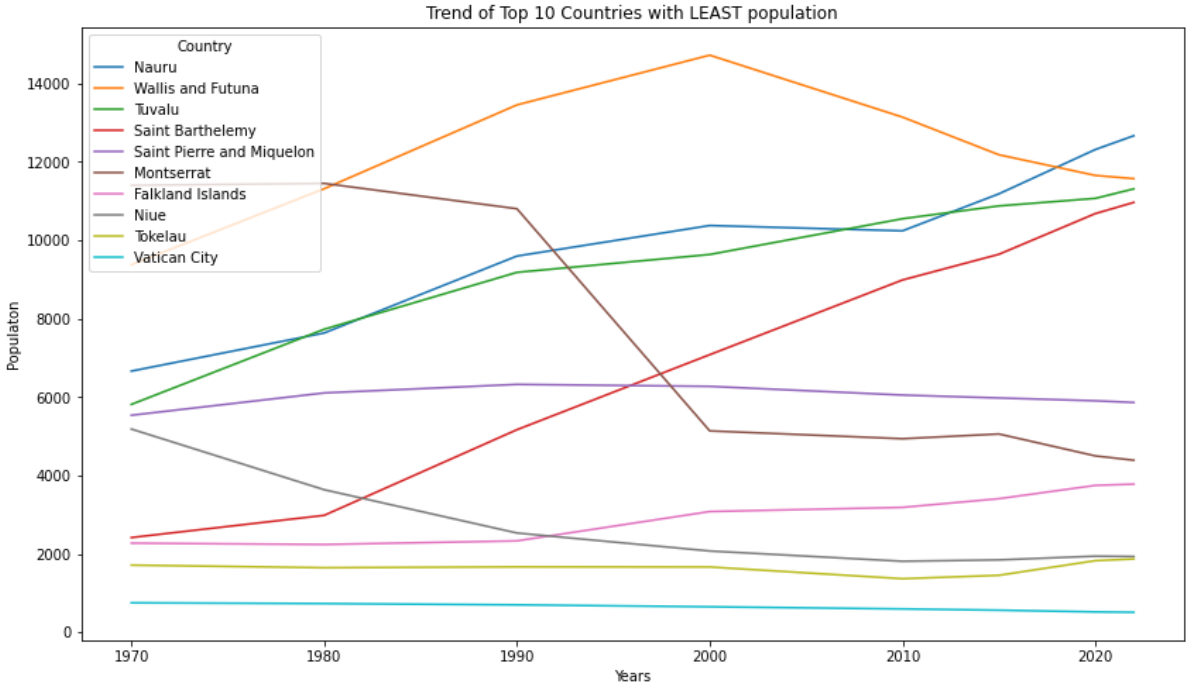
In [36]:
```
df_copy.sort_values(by='2022', ascending=True, inplace=True)

df_btm10 = df_copy['2022'].head(10)
df_btm10
```

Out[36]:
```
Country
Vatican City                    510
Tokelau                        1871
Niue                           1934
Falkland Islands               3780
Montserrat                     4390
Saint Pierre and Miquelon      5862
Saint Barthelemy              10967
Tuvalu                        11312
Wallis and Futuna             11572
Nauru                         12668
Name: 2022, dtype: int64
```

In [37]:
```
df_btm10.plot(kind='barh', figsize=(10, 10), color='darkred')
plt.xlabel('Population')
plt.title('Top 10 Countries With LEAST Population 2022')

plt.show()
```



In [38]:
```
#  top 10 countries with least population trend.

inplace = True
df_copy.sort_values(by='2022', ascending=False, axis=0, inplace=Tru

df_bttm10 = df_copy.tail(10)

df_bttm10 = df_bttm10[years].transpose()

df_bttm10.index = df_bttm10.index.map(int)
df_bttm10.plot(kind='line', figsize=(14, 8))

plt.title('Trend of Top 10 Countries with LEAST population')
plt.ylabel('Populaton')
plt.xlabel('Years')
plt.show()
```



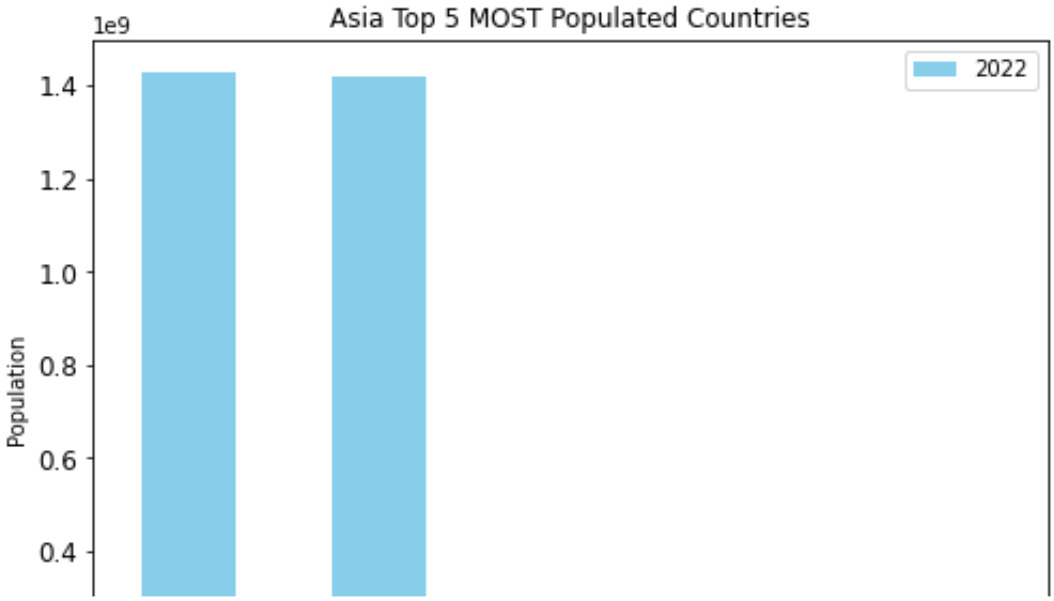
Top 5 Most Populated Countries By Continents

Here, we don't need to copy our dataframe or use the copied dataframe. We will use our original dataframe 'df' in this section

In [39]:
```
# creating dataframes for countries per continent

# Asia
asian_countries = df.loc[df["Continent"]=="Asia"].sort_values(by=["

# Africa
african_countries = df.loc[df["Continent"]=="Africa"].sort_values(b

# Europe
european_countries = df.loc[df["Continent"]=="Europe"].sort_values(

# North America
na_countries = df.loc[df["Continent"]=="North America"].sort_values

# Oceania
oc_countries = df.loc[df["Continent"]=="Oceania"].sort_values(by=["

# South America
sa_countries = df.loc[df["Continent"]=="South America"].sort_values
```

In [40]:
```
# plotting top 5 MOST populated countries by continent

# Asian countries
asian_countries[["Country", "2022"]].sort_values(by="2022", ascendi

# African countries
african_countries[["Country", "2022"]].sort_values(by="2022", ascen

# European countries
european_countries[["Country", "2022"]].sort_values(by="2022", asce

# North American countries
na_countries[["Country", "2022"]].sort_values(by="2022", ascending=

# Oceanian countries
oc_countries[["Country", "2022"]].sort_values(by="2022", ascending=

# South American countries
sa_countries[["Country", "2022"]].sort_values(by="2022", ascending=
```

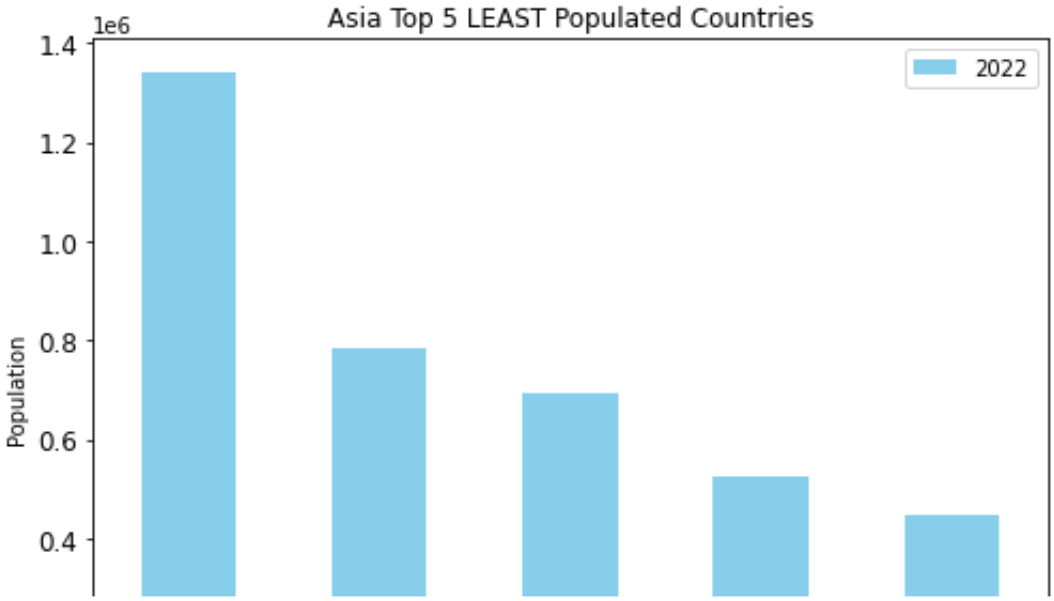
Out[40]:
```
<AxesSubplot:title={'center':'South America Top 5 MOST Populated C
ountries'}, xlabel='Country', ylabel='Population'>
```



Top 5 Least Populated Countries By Continents

In [41]:
```
lotting top 5 LEAST populated countries by continent

sian countries
an_countries[["Country", "2022"]].sort_values(by="2022", ascending=F

frican countries
ican_countries[["Country", "2022"]].sort_values(by="2022", ascending

uropean countries
opean_countries[["Country", "2022"]].sort_values(by="2022", ascendin

orth American countries
countries[["Country", "2022"]].sort_values(by="2022", ascending=Fals

ceanian countries
countries[["Country", "2022"]].sort_values(by="2022", ascending=Fals

outh American countries
countries[["Country", "2022"]].sort_values(by="2022", ascending=Fals
```

Out[41]:
```
<AxesSubplot:title={'center':'South America Top 5 LEAST Populated
Countries'}, xlabel='Country', ylabel='Population'>
```



Population Projection

World 2030 Population Projection

In [42]:
```
# current world population

df['2022'].sum()
```

Out[42]: 7973413042

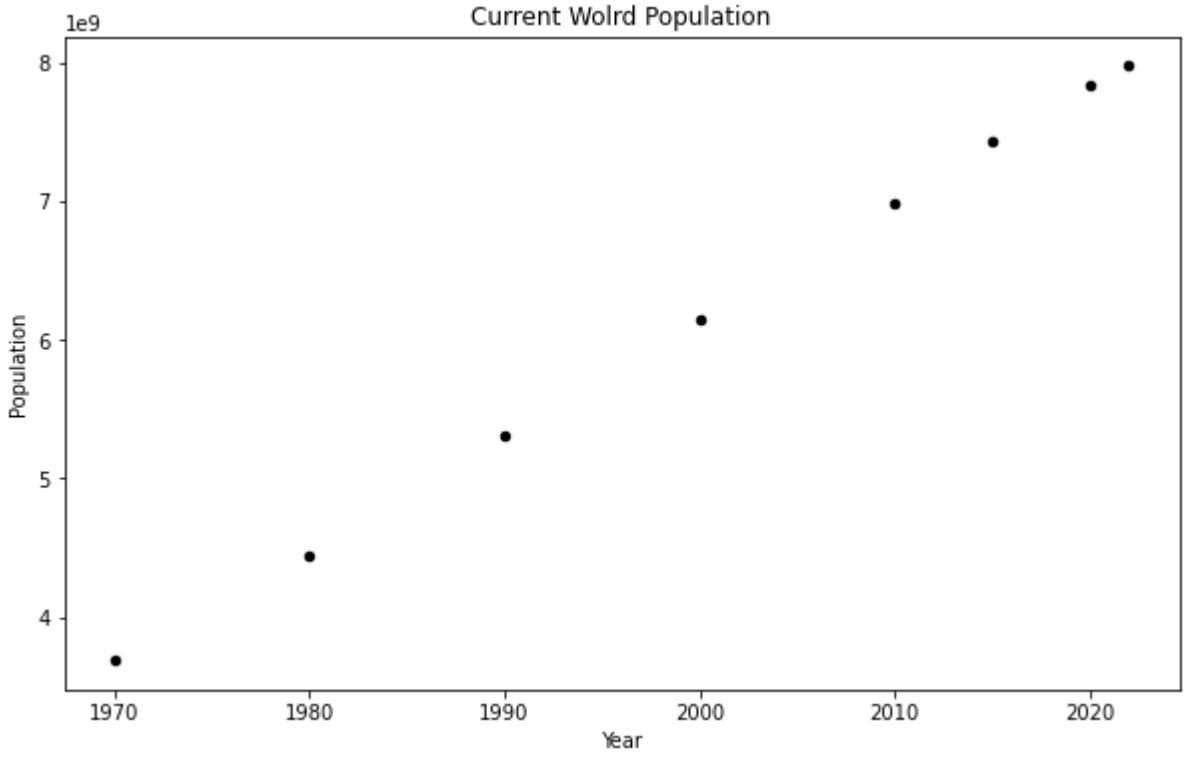
World current population: 7.9 billion

In [43]:
```
# relationship betewen years and total population, we will convert

df_tot = pd.DataFrame(df[years].sum(axis=0)) # use the sum() method

df_tot.index = map(int, df_tot.index) # change the years to type in

df_tot.reset_index(inplace = True) # reset the index to put in back

df_tot.columns = ['year', 'total'] # rename columns

df_tot.head()
```

Out[43]:

	year	total
0	1970	3694136661
1	1980	4442400371
2	1990	5314191665
3	2000	6147055703
4	2010	6983784998

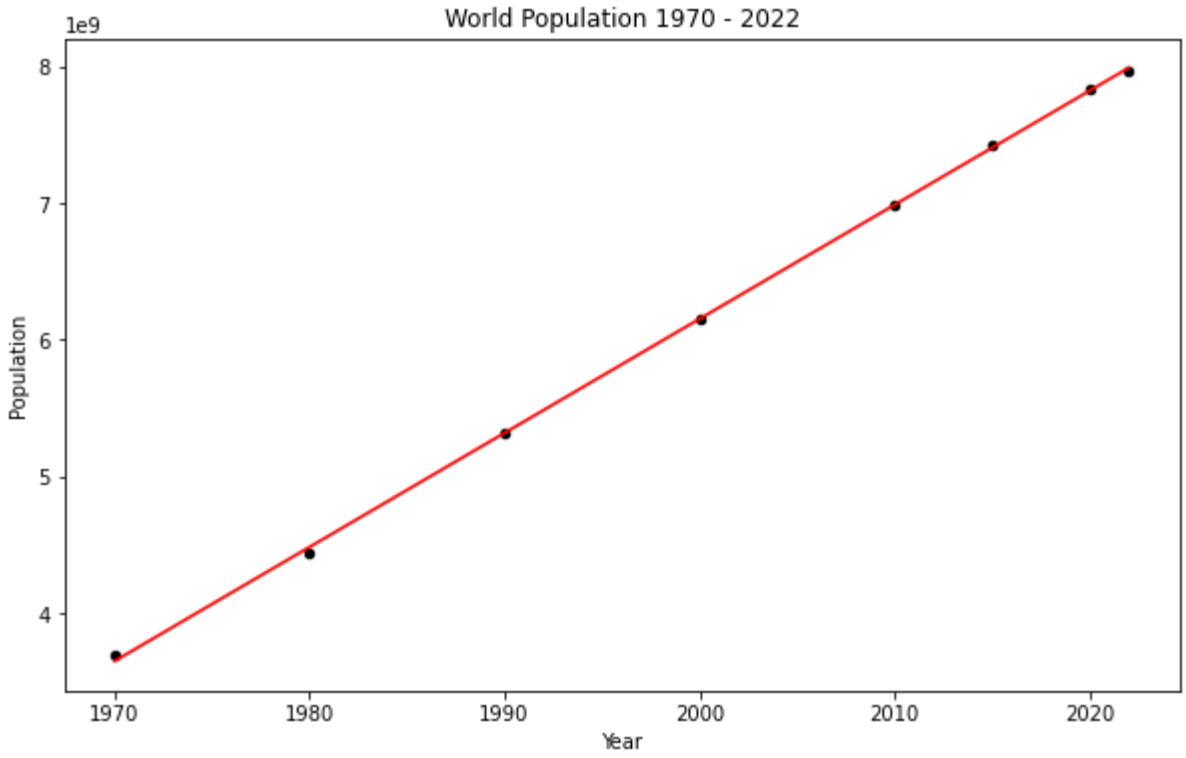
In [44]:
```
# plotting a scatter plot for year vs total population

df_tot.plot(kind='scatter', x='year', y='total', figsize=(10, 6), c

plt.title('Current Wolrd Population')
plt.xlabel('Year')
plt.ylabel('Population')

plt.show()
```



In [45]:
```
# fitting our data

x = df_tot['year']
y = df_tot['total']
fit = np.polyfit(x, y, deg=1)

fit
```

Out[45]:
```
array([ 8.33710451e+07, -1.60587660e+11])
```

In [46]:
```
# plotting the regression line on the scatter plot

df_tot.plot(kind='scatter', x='year', y='total', figsize=(10, 6), c

plt.title('World Population 1970 - 2022')
plt.xlabel('Year')
plt.ylabel('Population')

# plot line of best fit
plt.plot(x, fit[0] * x + fit[1], color='red') # recall that x is th
plt.annotate('y={0:.0f} x + {1:.0f}'.format(fit[0], fit[1]), xy=(20

plt.show()

# print out the line of best fit
'World Population = {0:.0f} * Year + {1:.0f}'.format(fit[0], fit[1]
```



Out[46]: 'World Population = 83371045 * Year + -160587659522'

Population Projection By Continents

Asia 2030 Population Projection

In [47]:
```
# current asia population

asian_countries['2022'].sum()
```

Out[47]: 4721383274

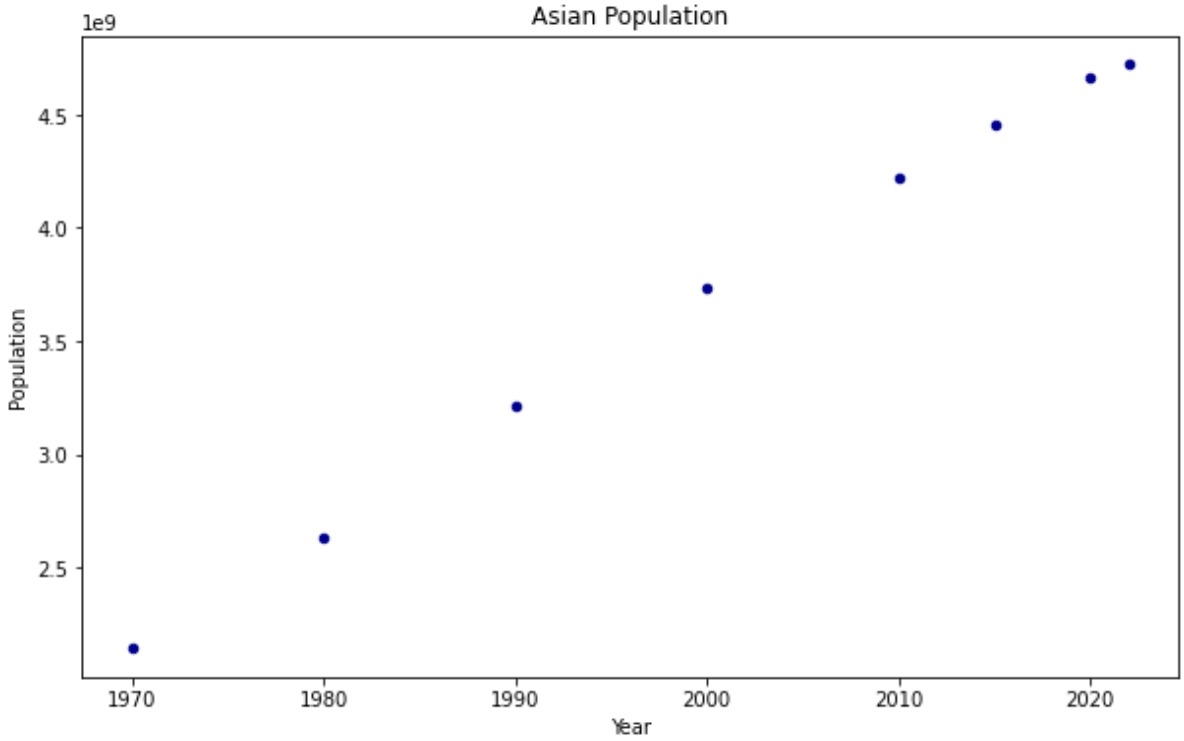
Asia current population: 4.7 billion

In [48]:
```
df_a230 = df[(df['Continent'] == 'Asia')]

# relationship betewen years and total asia population, we will con

df_asia_tot = pd.DataFrame(df_a230[years].sum(axis=0))

df_asia_tot.index = map(int, df_asia_tot.index)

df_asia_tot.reset_index(inplace = True)

df_asia_tot.columns = ['year', 'total']

df_asia_tot.tail()
```

Out[48]:

	year	total
3	2000	3735089604
4	2010	4220041327
5	2015	4458250182
6	2020	4663086535
7	2022	4721383274

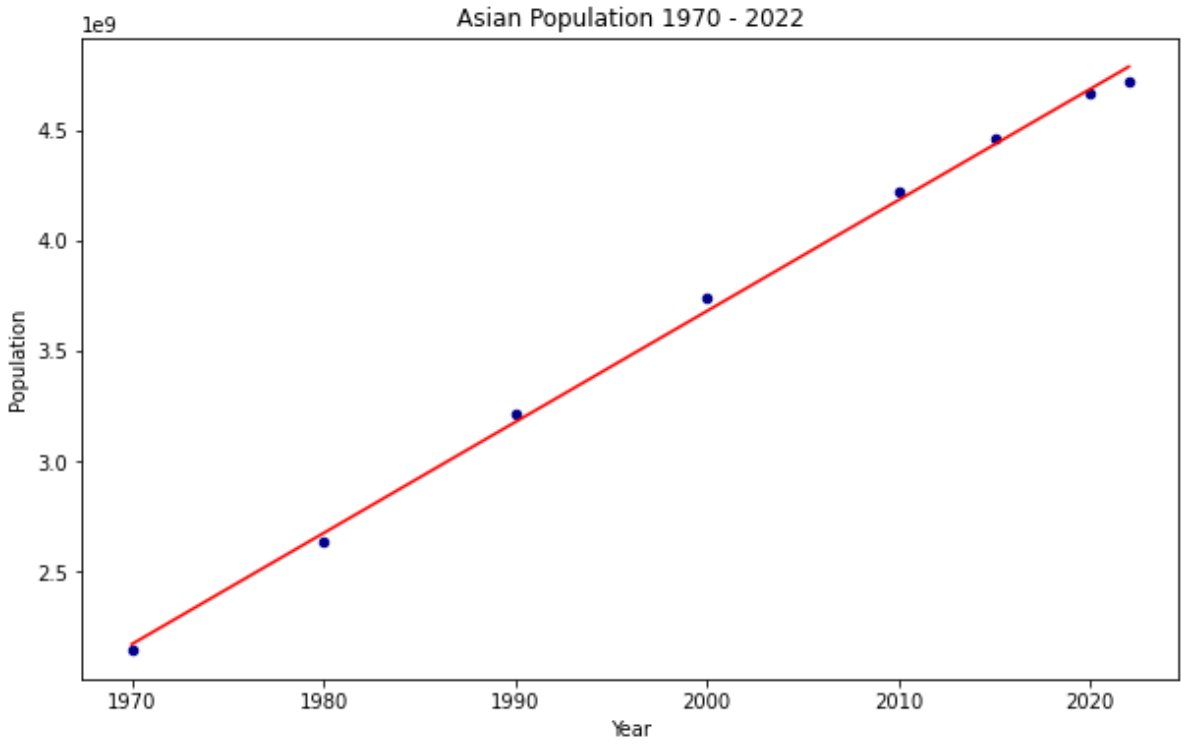
In [49]:
```
df_asia_tot.plot(kind='scatter', x='year', y='total', figsize=(10,

plt.title('Asian Population')
plt.xlabel('Year')
plt.ylabel('Population')

plt.show()
```



In [50]:
```
# fitting asian data

x_as = df_asia_tot['year']
y_as = df_asia_tot['total']
fit_as = np.polyfit(x_as, y_as, deg=1)

fit_as
```

Out[50]:
```
array([ 5.03219504e+07, -9.69643506e+10])
```

In [51]:
```
# plotting Asian Populaion regression line on the scatter plot

df_asia_tot.plot(kind='scatter', x='year', y='total', figsize=(10,

plt.title('Asian Population 1970 - 2022')
plt.xlabel('Year')
plt.ylabel('Population')

# plot line of best fit
plt.plot(x_as, fit_as[0] * x_as + fit_as[1], color='red') # recall
plt.annotate('y={0:.0f} x + {1:.0f}'.format(fit_as[0], fit_as[1]),

plt.show()

# print out the line of best fit
'Asian Population = {0:.0f} * Year + {1:.0f}'.format(fit_as[0], fit
```



Out[51]: 'Asian Population = 50321950 * Year + -96964350561'

Africa 2030 Population Projection

In [52]:
```
# current africa population

african_countries['2022'].sum()
```

Out[52]: 1426730932

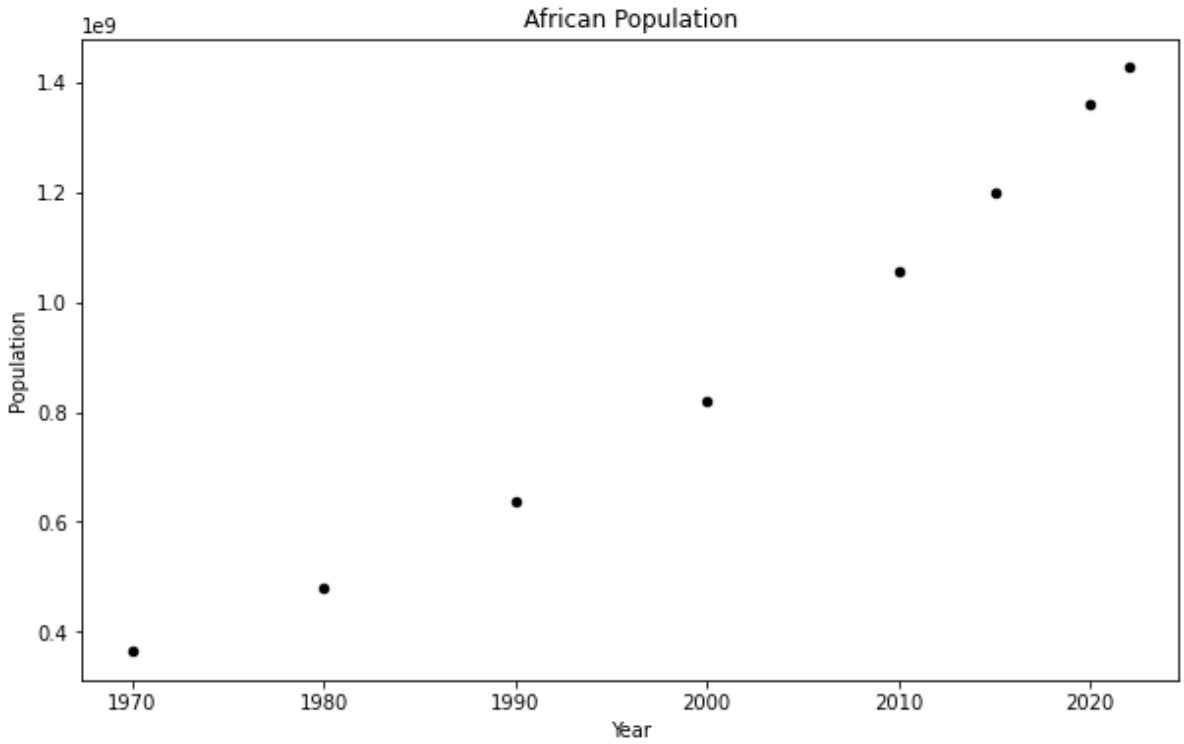
Africa current population: 1.4 billion

In [53]:
```
df_af230 = df[(df['Continent'] == 'Africa')]

# relationship betewen years and total asia population, we will con

df_afri_tot = pd.DataFrame(df_af230[years].sum(axis=0))

df_afri_tot.index = map(int, df_afri_tot.index)

df_afri_tot.reset_index(inplace = True)

df_afri_tot.columns = ['year', 'total']

df_afri_tot.tail()
```

Out[53]:

	year	total
3	2000	818946032
4	2010	1055228072
5	2015	1201102442
6	2020	1360671810
7	2022	1426730932

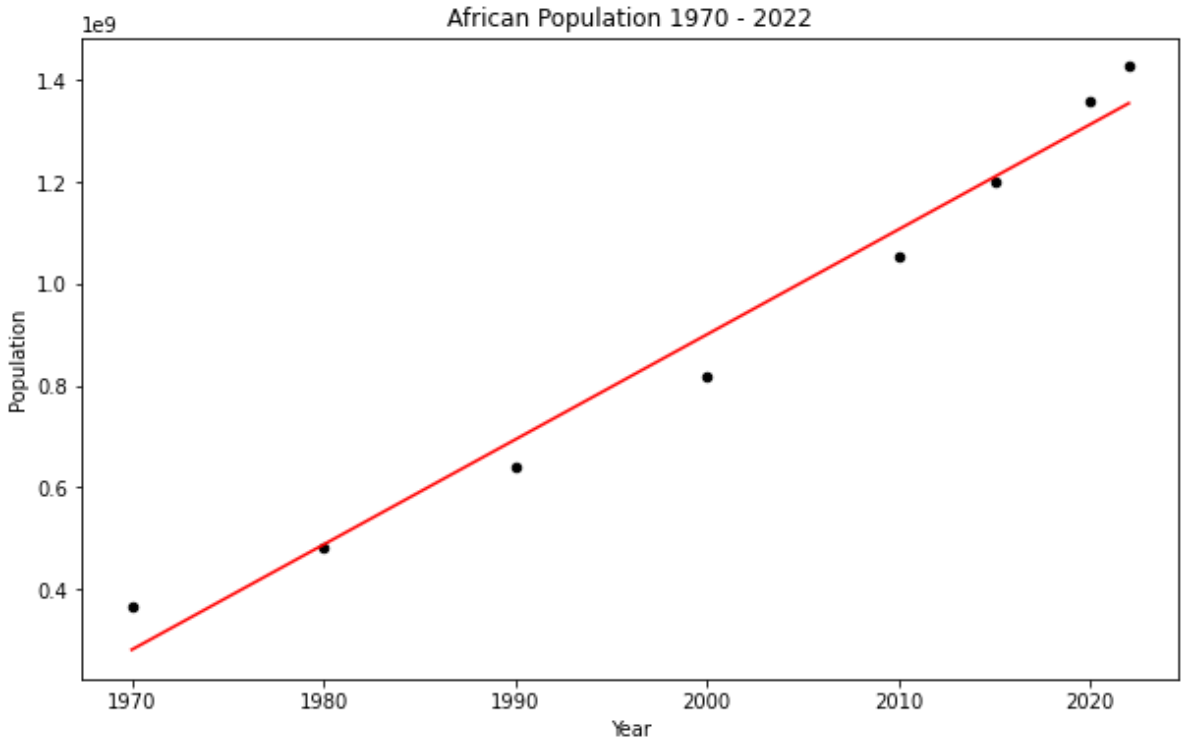
In [54]:
```
df_afri_tot.plot(kind='scatter', x='year', y='total', figsize=(10,

plt.title('African Population')
plt.xlabel('Year')
plt.ylabel('Population')

plt.show()
```



In [55]:
```
# fitting african data

x_af = df_afri_tot['year']
y_af = df_afri_tot['total']
fit_af = np.polyfit(x_af, y_af, deg=1)

fit_af
```

Out[55]:
```
array([ 2.06561727e+07, -4.04119432e+10])
```

In [56]:
```
# plotting African Populaion regression line on the scatter plot

df_afri_tot.plot(kind='scatter', x='year', y='total', figsize=(10,

plt.title('African Population 1970 - 2022')
plt.xlabel('Year')
plt.ylabel('Population')

# plot line of best fit
plt.plot(x_af, fit_af[0] * x_af + fit_af[1], color='red') # recall
plt.annotate('y={0:.0f} x + {1:.0f}'.format(fit_af[0], fit_af[1]),

plt.show()

# print out the line of best fit
'African Population = {0:.0f} * Year + {1:.0f}'.format(fit_af[0], f
```



Out[56]:
```
'African Population = 20656173 * Year + -40411943187'
```

Europe 2030 Population Projection

In [57]:
```
# current europe population

european_countries['2022'].sum()
```

Out[57]:
```
743147538
```

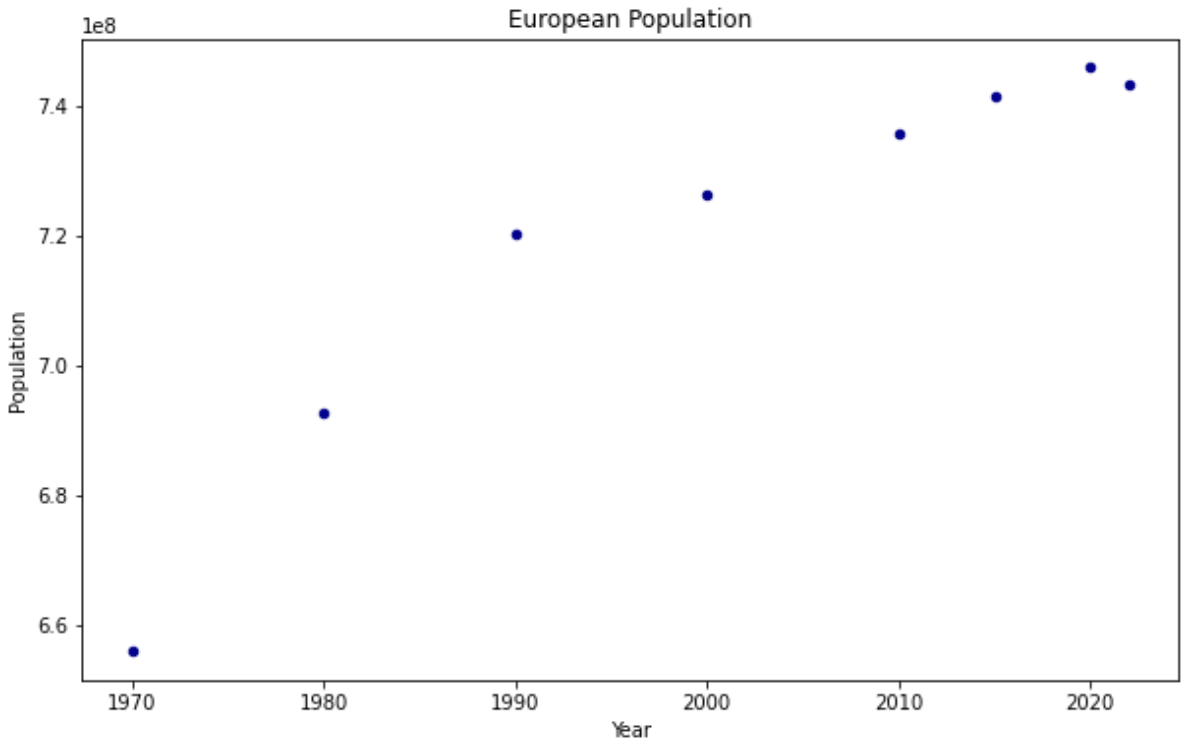
Europe current population: 743 million

In [58]:
```
df_eu230 = df[(df['Continent'] == 'Europe')]

# relationship betewen years and total asia population, we will con

df_eu_tot = pd.DataFrame(df_eu230[years].sum(axis=0))

df_eu_tot.index = map(int, df_eu_tot.index)

df_eu_tot.reset_index(inplace = True)

df_eu_tot.columns = ['year', 'total']

df_eu_tot.tail()
```

Out[58]:

	year	total
3	2000	726093423
4	2010	735613934
5	2015	741535608
6	2020	745792196
7	2022	743147538

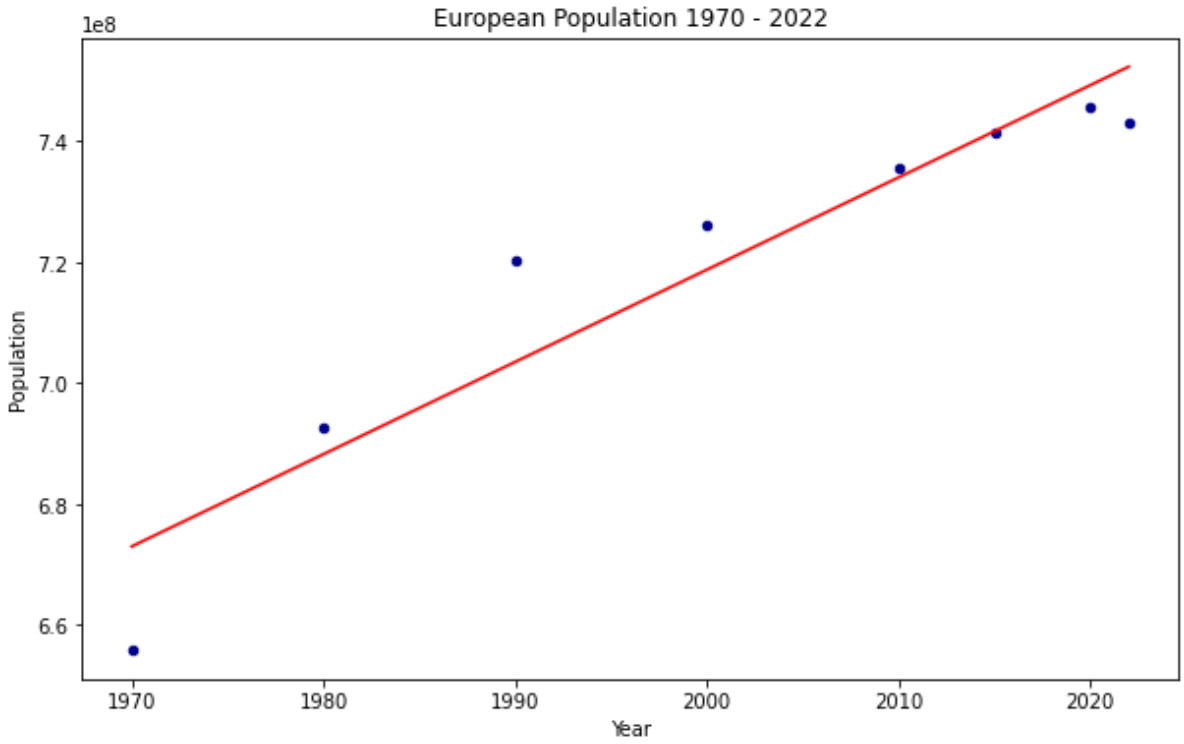
In [59]:
```
df_eu_tot.plot(kind='scatter', x='year', y='total', figsize=(10, 6)

plt.title('European Population')
plt.xlabel('Year')
plt.ylabel('Population')

plt.show()
```



In [60]:
```
# fitting european data

x_eu = df_eu_tot['year']
y_eu = df_eu_tot['total']
fit_eu = np.polyfit(x_eu, y_eu, deg=1)

fit_eu
```

Out[60]:
```
array([ 1.52598839e+06, -2.33319268e+09])
```

In [61]:
```
# plotting European Populaion regression line on the scatter plot

df_eu_tot.plot(kind='scatter', x='year', y='total', figsize=(10, 6)

plt.title('European Population 1970 - 2022')
plt.xlabel('Year')
plt.ylabel('Population')

# plot line of best fit
plt.plot(x_eu, fit_eu[0] * x_eu + fit_eu[1], color='red') # recall
plt.annotate('y={0:.0f} x + {1:.0f}'.format(fit_eu[0], fit_eu[1]),

plt.show()

# print out the line of best fit
'European Population = {0:.0f} * Year + {1:.0f}'.format(fit_eu[0],
```



Out[61]: 'European Population = 1525988 * Year + -2333192682'

North America 2030 Population Projection

In [62]:
```
# current north america population

na_countries['2022'].sum()
```

Out[62]: 600296136

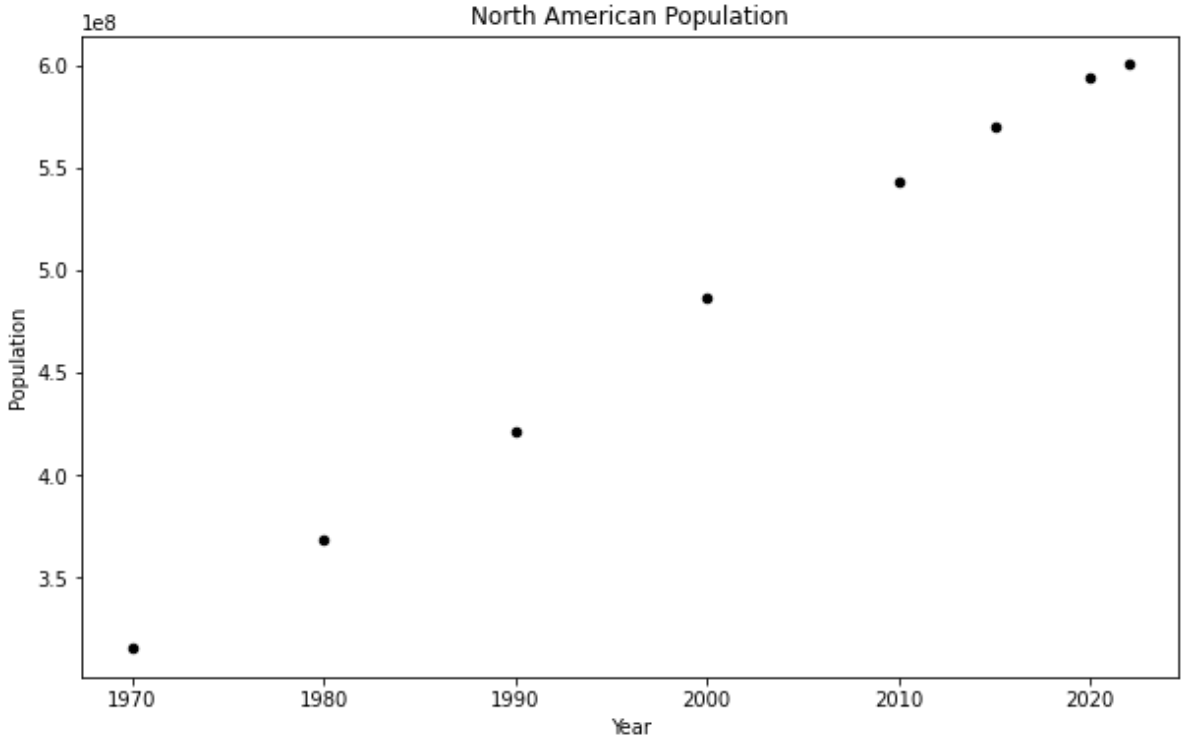
North America current population: 600 million

In [63]:
```
df_na230 = df[(df['Continent'] == 'North America')]

# relationship betewen years and total asia population, we will con

df_na_tot = pd.DataFrame(df_na230[years].sum(axis=0))

df_na_tot.index = map(int, df_na_tot.index)

df_na_tot.reset_index(inplace = True)

df_na_tot.columns = ['year', 'total']

df_na_tot.tail()
```

Out[63]:

	year	total
3	2000	486069584
4	2010	542720651
5	2015	570383850
6	2020	594236593
7	2022	600296136

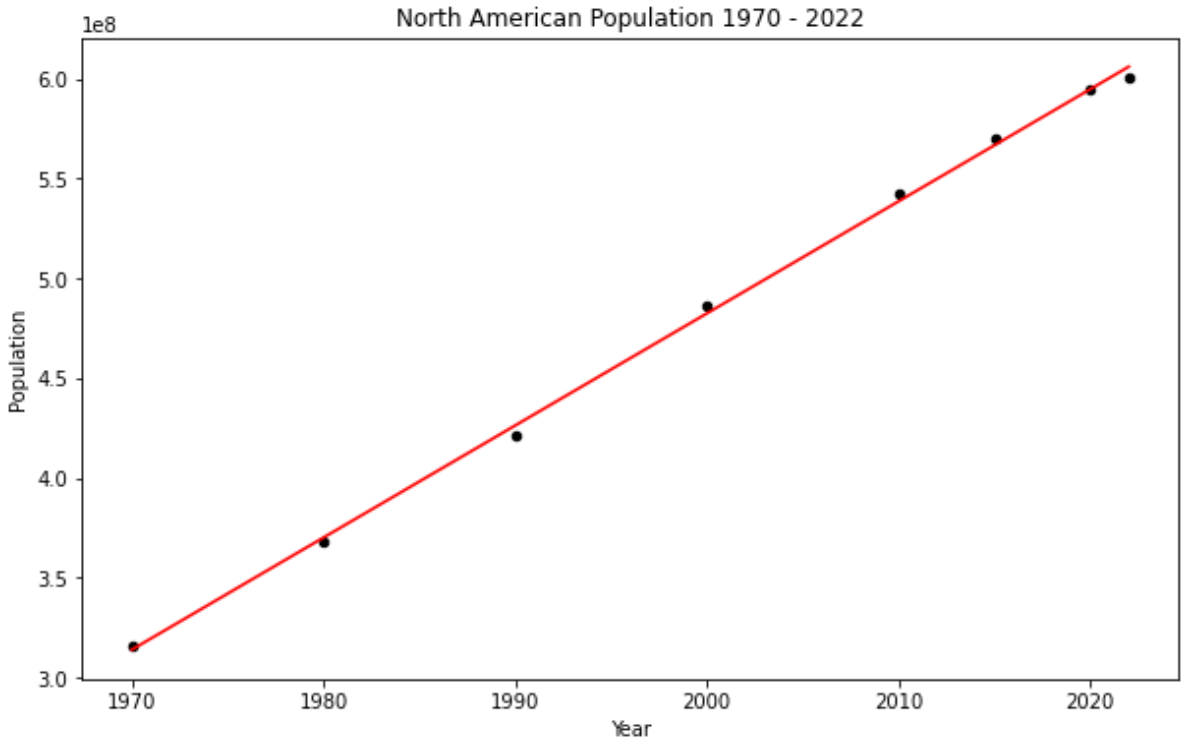
In [64]:
```
df_na_tot.plot(kind='scatter', x='year', y='total', figsize=(10, 6)

plt.title('North American Population')
plt.xlabel('Year')
plt.ylabel('Population')

plt.show()
```



In [65]:
```
# fitting north american data

x_na = df_na_tot['year']
y_na = df_na_tot['total']
fit_na = np.polyfit(x_na, y_na, deg=1)

fit_na
```

Out[65]:
```
array([ 5.61735693e+06, -1.07522914e+10])
```

In [66]:
```
# plotting North American Populaion regression line on the scatter

df_na_tot.plot(kind='scatter', x='year', y='total', figsize=(10, 6)

plt.title('North American Population 1970 - 2022')
plt.xlabel('Year')
plt.ylabel('Population')

# plot line of best fit
plt.plot(x_na, fit_na[0] * x_na + fit_na[1], color='red') # recall
plt.annotate('y={0:.0f} x + {1:.0f}'.format(fit_na[0], fit_na[1]),

plt.show()

# print out the line of best fit
'North American Population = {0:.0f} * Year + {1:.0f}'.format(fit_n
```



Out[66]: 'North American Population = 5617357 * Year + -10752291387'

Oceania 2030 Population Projection

In [67]:
```
# current oceania population

oc_countries['2022'].sum()
```

Out[67]: 45038554

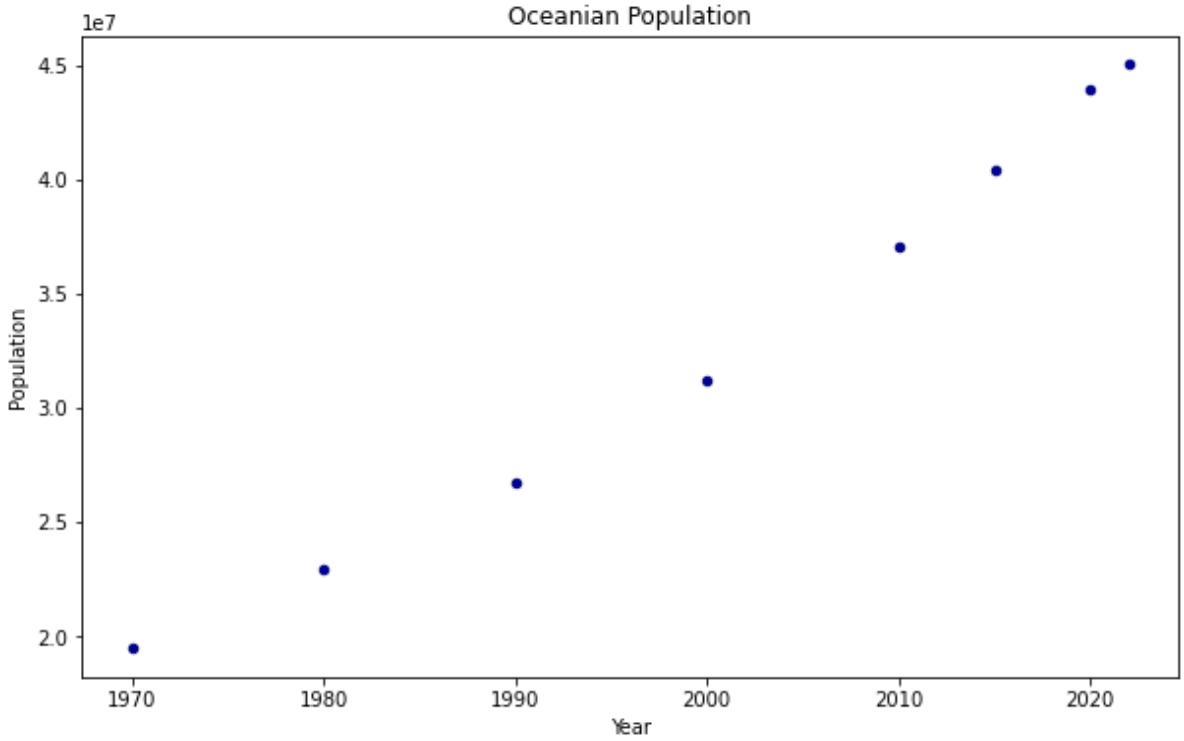
Oceania current population: 45 million

In [68]:
```
df_oc230 = df[(df['Continent'] == 'Oceania')]

# relationship betewen years and total asia population, we will con

df_oc_tot = pd.DataFrame(df_oc230[years].sum(axis=0))

df_oc_tot.index = map(int, df_oc_tot.index)

df_oc_tot.reset_index(inplace = True)

df_oc_tot.columns = ['year', 'total']

df_oc_tot.tail()
```

Out[68]:

	year	total
3	2000	31222778
4	2010	37102764
5	2015	40403283
6	2020	43933426
7	2022	45038554

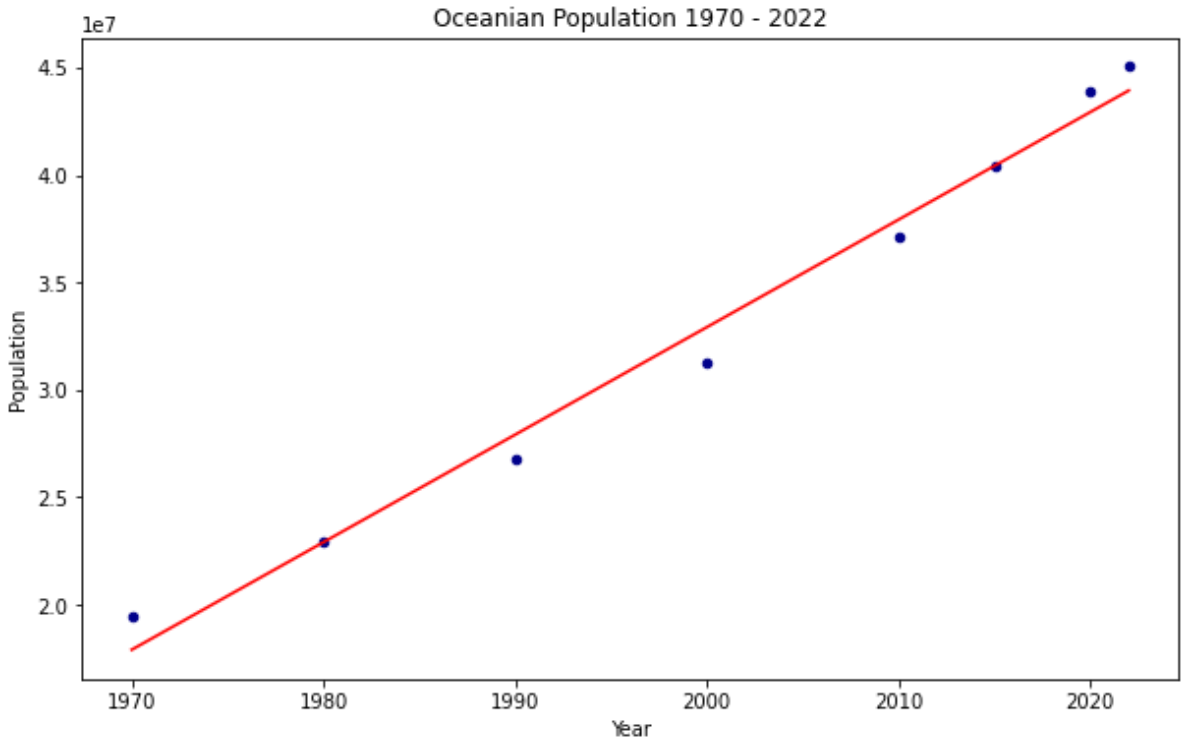
In [69]:
```
df_oc_tot.plot(kind='scatter', x='year', y='total', figsize=(10, 6)

plt.title('Oceanian Population')
plt.xlabel('Year')
plt.ylabel('Population')

plt.show()
```



In [70]:
```
# fitting oceania data

x_oc = df_oc_tot['year']
y_oc = df_oc_tot['total']
fit_oc = np.polyfit(x_oc, y_oc, deg=1)

fit_oc
```

Out[70]:
```
array([ 5.00543813e+05, -9.68169960e+08])
```

In [71]:
```
# plotting Oceanian Populaion regression line on the scatter plot

df_oc_tot.plot(kind='scatter', x='year', y='total', figsize=(10, 6)

plt.title('Oceanian Population 1970 - 2022')
plt.xlabel('Year')
plt.ylabel('Population')

# plot line of best fit
plt.plot(x_oc, fit_oc[0] * x_oc + fit_oc[1], color='red') # recall
plt.annotate('y={0:.0f} x + {1:.0f}'.format(fit_oc[0], fit_oc[1]),

plt.show()

# print out the line of best fit
'Oceanian Population = {0:.0f} * Year + {1:.0f}'.format(fit_oc[0],
```



Out[71]: 'Oceanian Population = 500544 * Year + -968169960'

South America 2030 Population Projection

In [72]:
```
# current south american population

sa_countries['2022'].sum()
```

Out[72]: 436816608

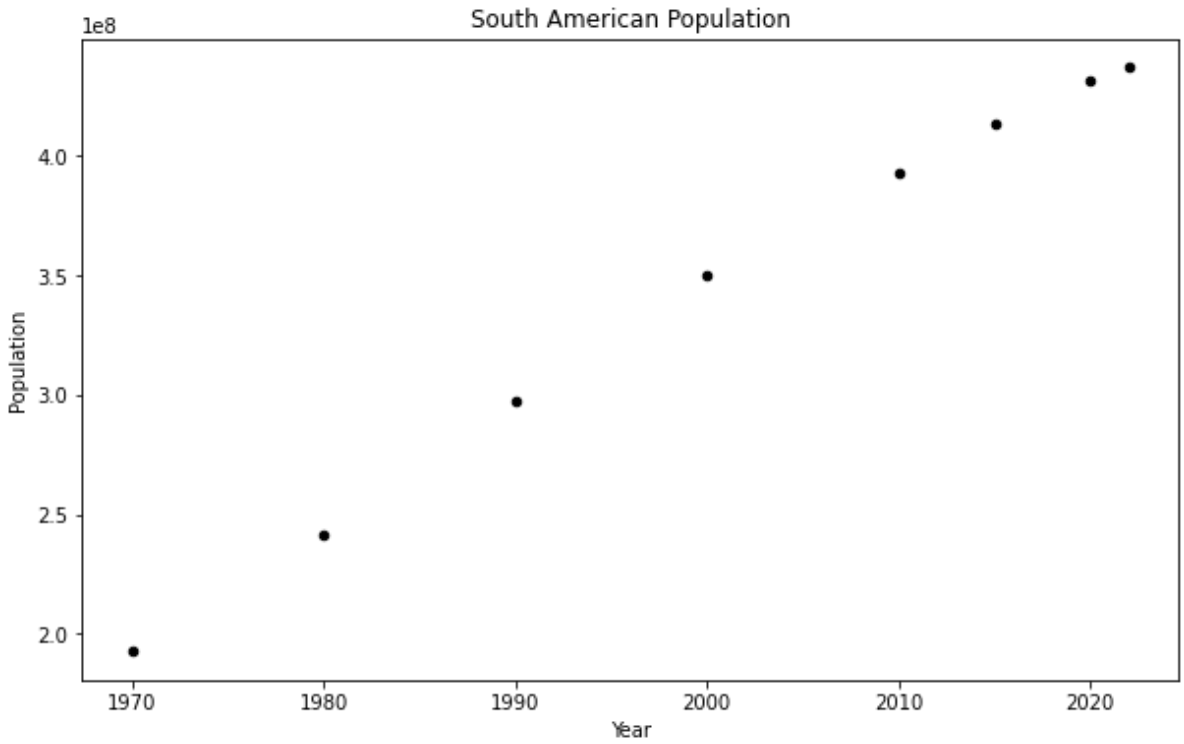
South America current population: 436 million

In [73]:
```
df_sa230 = df[(df['Continent'] == 'South America')]

# relationship betewen years and total asia population, we will con

df_sa_tot = pd.DataFrame(df_sa230[years].sum(axis=0))

df_sa_tot.index = map(int, df_sa_tot.index)

df_sa_tot.reset_index(inplace = True)

df_sa_tot.columns = ['year', 'total']

df_sa_tot.tail()
```

Out[73]:

	year	total
3	2000	349634282
4	2010	393078250
5	2015	413134396
6	2020	431530043
7	2022	436816608

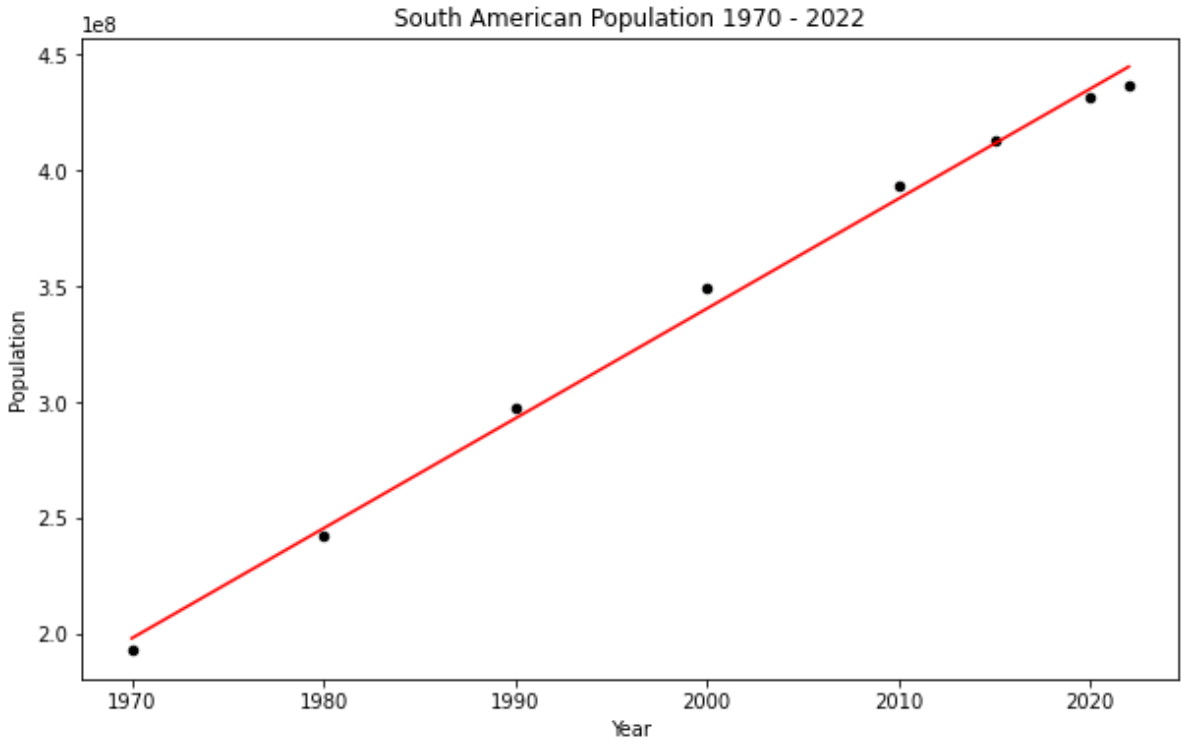
In [74]:
```
df_sa_tot.plot(kind='scatter', x='year', y='total', figsize=(10, 6)

plt.title('South American Population')
plt.xlabel('Year')
plt.ylabel('Population')

plt.show()
```



In [75]:
```
# fitting south america data

x_sa = df_sa_tot['year']
y_sa = df_sa_tot['total']
fit_sa = np.polyfit(x_sa, y_sa, deg=1)

fit_sa
```

Out[75]:
```
array([ 4.74903293e+06, -9.15771175e+09])
```

In [76]:
```
# plotting South American Populaion regression line on the scatter

df_sa_tot.plot(kind='scatter', x='year', y='total', figsize=(10, 6)

plt.title('South American Population 1970 - 2022')
plt.xlabel('Year')
plt.ylabel('Population')

# plot line of best fit
plt.plot(x_sa, fit_sa[0] * x_sa + fit_sa[1], color='red') # recall
plt.annotate('y={0:.0f} x + {1:.0f}'.format(fit_sa[0], fit_sa[1]),

plt.show()

# print out the line of best fit
'South American Population = {0:.0f} * Year + {1:.0f}'.format(fit_s
```



Out[76]: 'South American Population = 4749033 * Year + -9157711746'

Canada 2026 Population Census Projection

In [77]:
```
df_can = df[(df['Country'] == 'Canada')]

df_can['2022'].sum()
```

Out[77]: 38454327

Canada current population: 38 million

Last population census: 2021

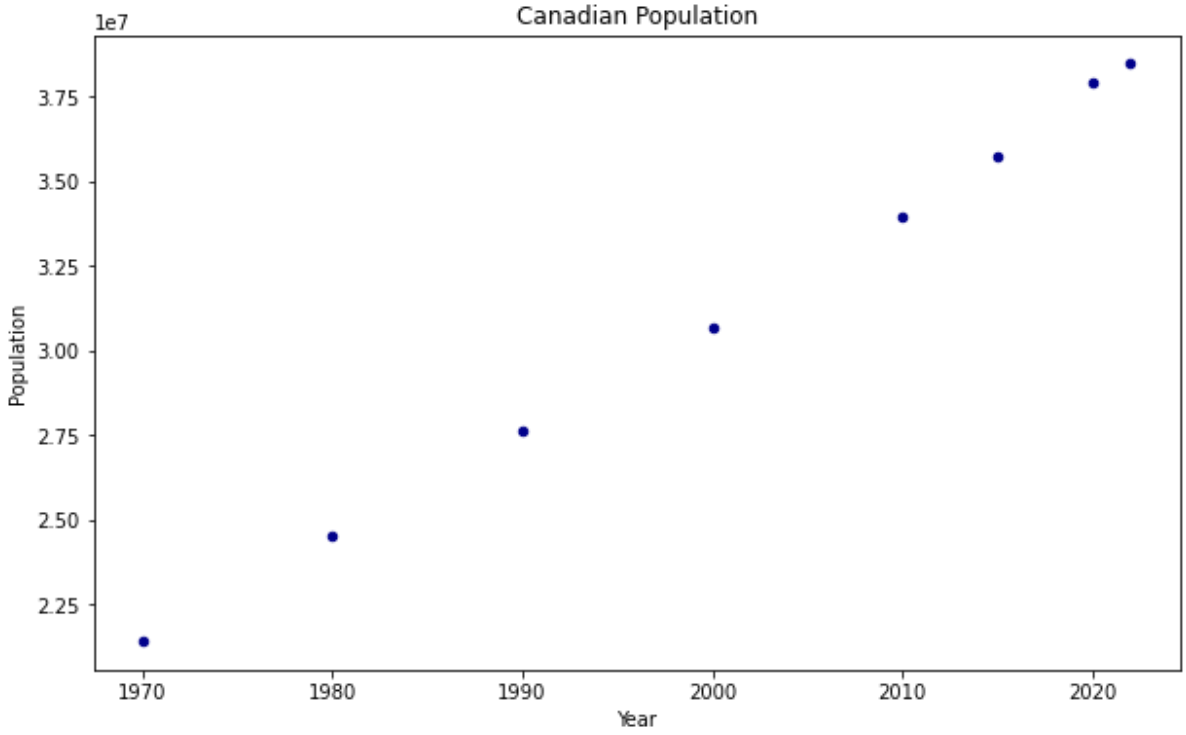
Next population census: 2026

In [78]:
```
# relationship betewen years and total population, we will convert
df_can_tot = pd.DataFrame(df_can[years].sum(axis=0))
df_can_tot.index = map(int, df_can_tot.index)
df_can_tot.reset_index(inplace = True)
df_can_tot.columns = ['year', 'total']
df_can_tot.tail()
```

Out[78]:

	year	total
3	2000	30683313
4	2010	33963412
5	2015	35732126
6	2020	37888705
7	2022	38454327

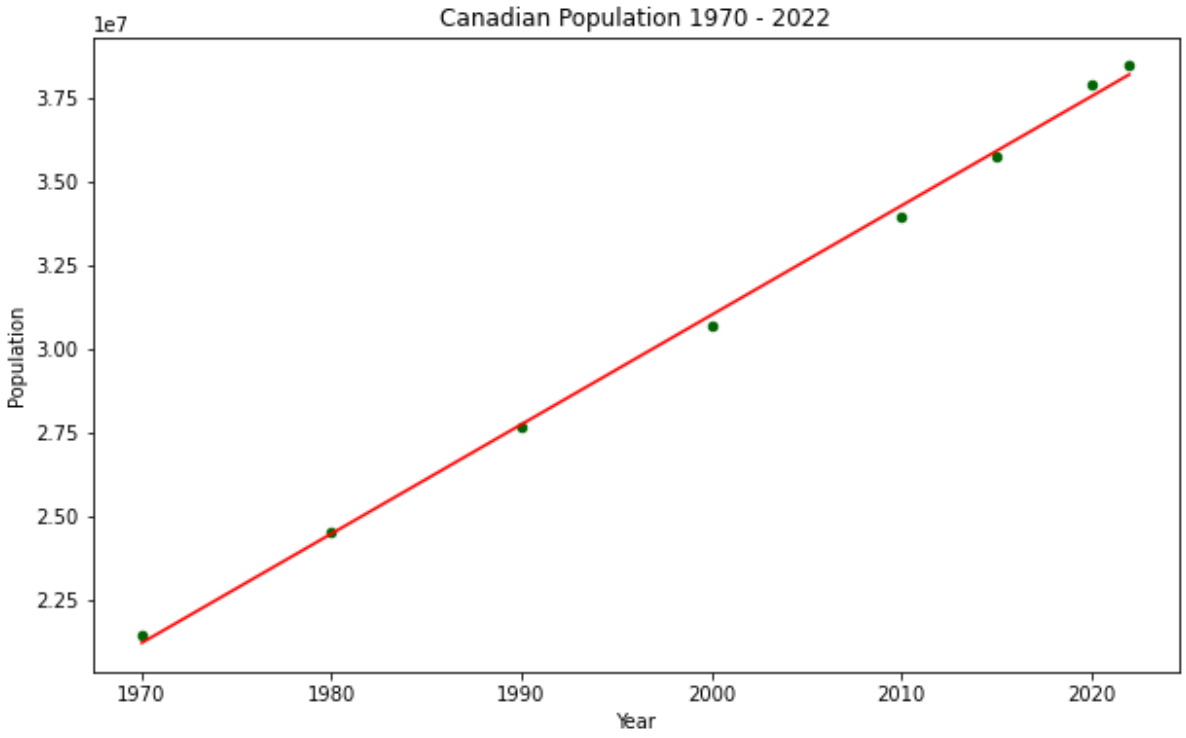
In [79]:
```
df_can_tot.plot(kind='scatter', x='year', y='total', figsize=(10, 6

plt.title('Canadian Population')
plt.xlabel('Year')
plt.ylabel('Population')

plt.show()
```



In [80]:
```
x_can = df_can_tot['year']
y_can = df_can_tot['total']
fit_can = np.polyfit(x_can, y_can, deg=1)

fit_can
```

Out[80]:
```
array([ 3.26758383e+05, -6.22512033e+08])
```

In [81]:
```
# plot the Canadian Populaion regression line on the scatter plot

df_can_tot.plot(kind='scatter', x='year', y='total', figsize=(10, 6

plt.title('Canadian Population 1970 - 2022')
plt.xlabel('Year')
plt.ylabel('Population')

# plot line of best fit
plt.plot(x_can, fit_can[0] * x_can + fit_can[1], color='red') # rec
plt.annotate('y={0:.0f} x + {1:.0f}'.format(fit_can[0], fit_can[1])

plt.show()

# print out the line of best fit
'Canadian Population = {0:.0f} * Year + {1:.0f}'.format(fit_can[0],
```



Out[81]: 'Canadian Population = 326758 * Year + -622512033'

Ireland 2027 Population Census Projection

In [82]:
```
df_ir = df[(df['Country'] == 'Ireland')]

df_ir['2022'].sum()
```

Out[82]: 5023109

Ireland current population: 5 million

Last population census: 2022

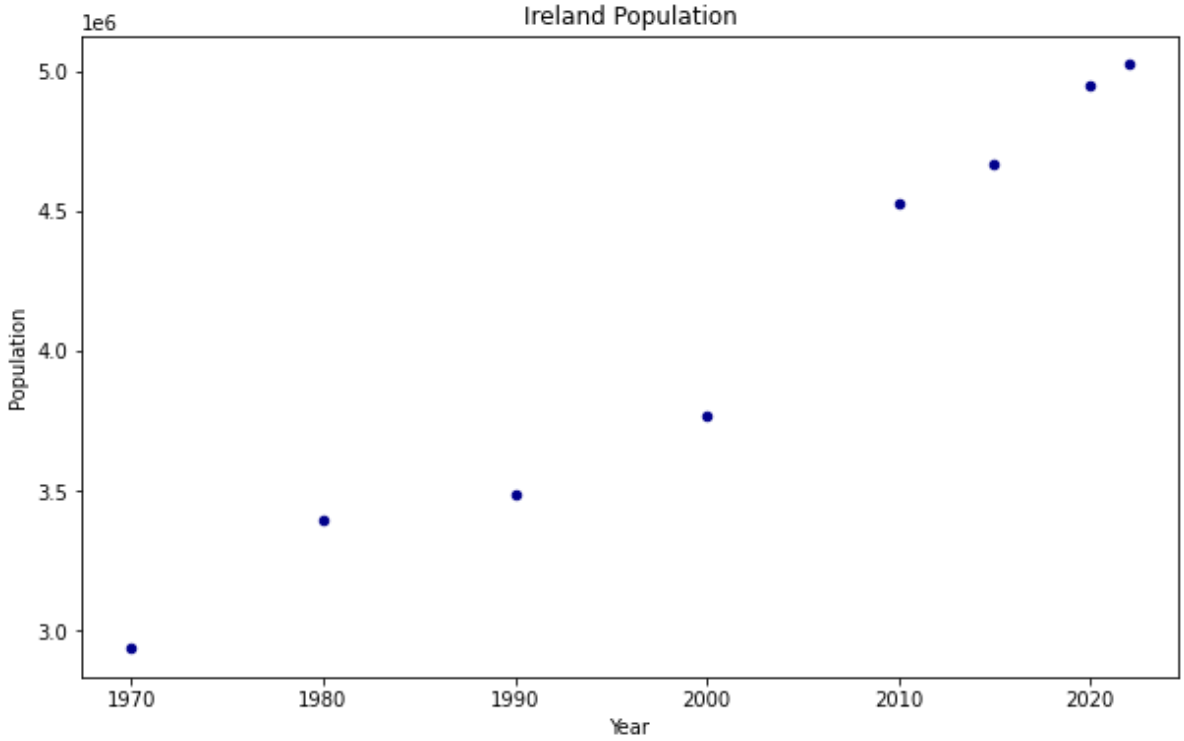
Next population census: 2027

In [83]:
```
# relationship betewen years and total population, we will convert
df_ir_tot = pd.DataFrame(df_ir[years].sum(axis=0))
df_ir_tot.index = map(int, df_ir_tot.index)
df_ir_tot.reset_index(inplace = True)
df_ir_tot.columns = ['year', 'total']
df_ir_tot.tail()
```

Out[83]:

	year	total
3	2000	3768950
4	2010	4524585
5	2015	4665760
6	2020	4946119
7	2022	5023109

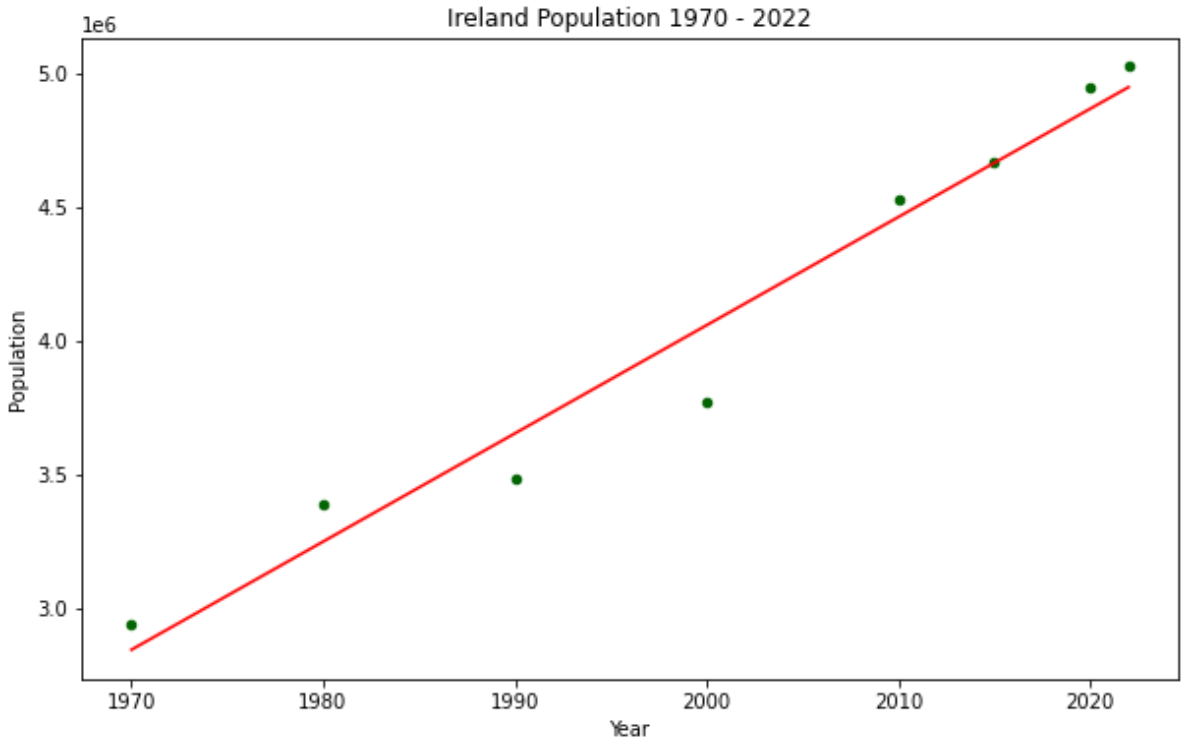
In [84]:
```
df_ir_tot.plot(kind='scatter', x='year', y='total', figsize=(10, 6)

plt.title('Ireland Population')
plt.xlabel('Year')
plt.ylabel('Population')

plt.show()
```



In [85]:
```
x_ir = df_ir_tot['year']
y_ir = df_ir_tot['total']
fit_ir = np.polyfit(x_ir, y_ir, deg=1)

fit_ir
```

Out[85]:
```
array([ 4.04177619e+04, -7.67780243e+07])
```

In [86]:
```
# plot the Ireland Populaion regression line on the scatter plot

df_ir_tot.plot(kind='scatter', x='year', y='total', figsize=(10, 6)

plt.title('Ireland Population 1970 - 2022')
plt.xlabel('Year')
plt.ylabel('Population')

# plot line of best fit
plt.plot(x_ir, fit_ir[0] * x_ir + fit_ir[1], color='red') # recall
plt.annotate('y={0:.0f} x + {1:.0f}'.format(fit_ir[0], fit_ir[1]),

plt.show()

# print out the line of best fit
'Ireland Population = {0:.0f} * Year + {1:.0f}'.format(fit_ir[0], f
```



Out[86]: 'Ireland Population = 40418 * Year + -76778024'

Australia 2026 Population Census Projection

In [87]:
```
df_au = df[(df['Country'] == 'Australia')]

df_au['2022'].sum()
```

Out[87]: 26177413

Australia current population: 26 million

Last population census: 2021

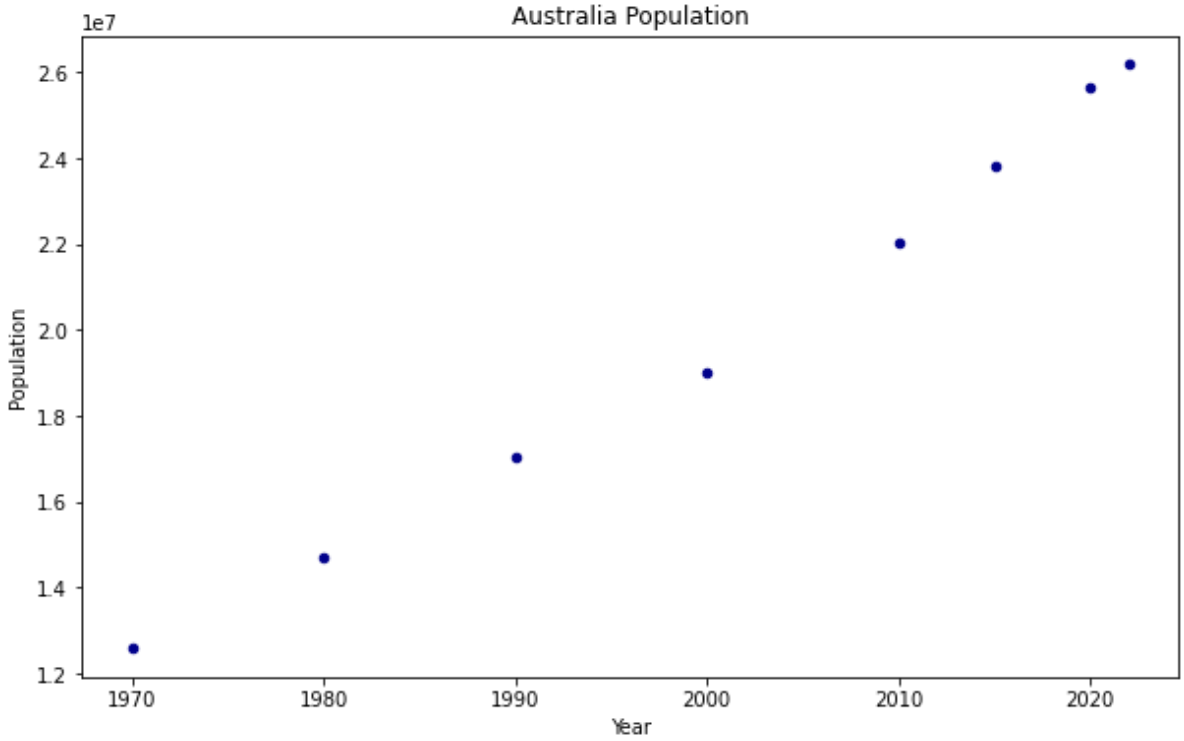
Next population census: 2026

In [88]:
```
# relationship betewen years and total population, we will convert
df_au_tot = pd.DataFrame(df_au[years].sum(axis=0))
df_au_tot.index = map(int, df_au_tot.index)
df_au_tot.reset_index(inplace = True)
df_au_tot.columns = ['year', 'total']
df_au_tot.tail()
```

Out[88]:

	year	total
3	2000	19017963
4	2010	22019168
5	2015	23820236
6	2020	25670051
7	2022	26177413

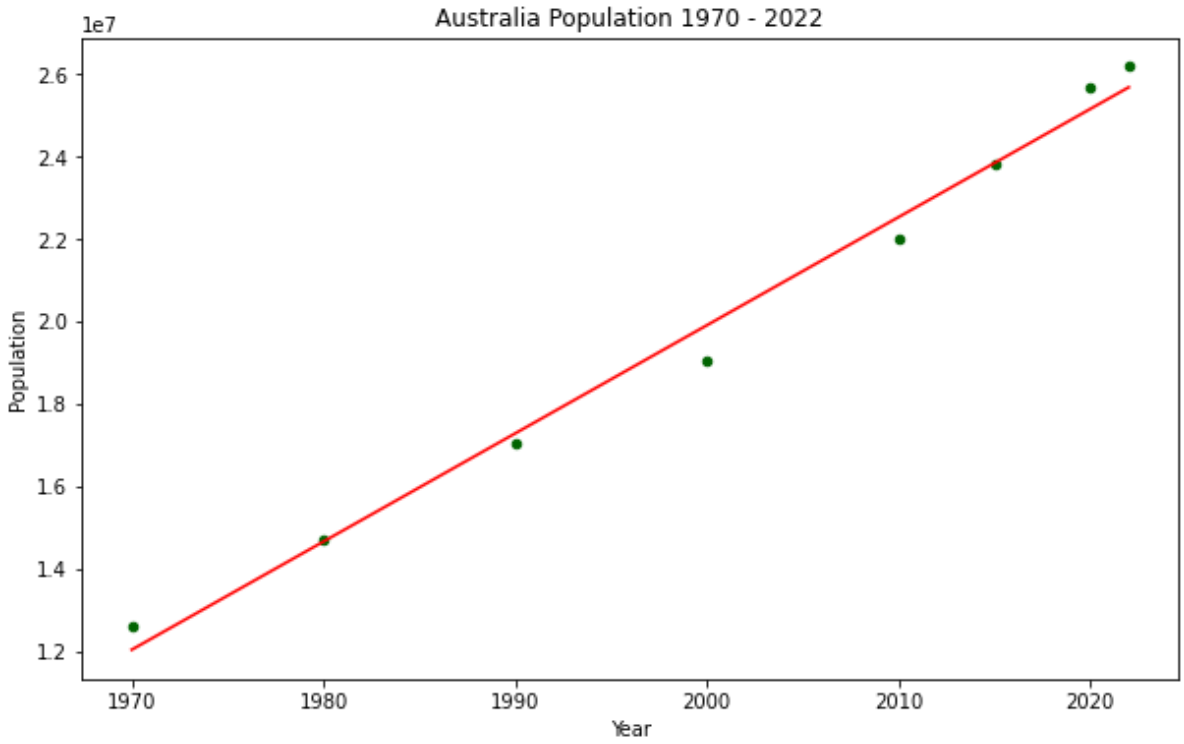
In [89]:
```
df_au_tot.plot(kind='scatter', x='year', y='total', figsize=(10, 6)

plt.title('Australia Population')
plt.xlabel('Year')
plt.ylabel('Population')

plt.show()
```



In [90]:
```
x_au = df_au_tot['year']
y_au = df_au_tot['total']
fit_au = np.polyfit(x_au, y_au, deg=1)

fit_au
```

Out[90]:
```
array([ 2.62563289e+05, -5.05224546e+08])
```

In [91]:
```
# plot the Australia Populaion regression line on the scatter plot

df_au_tot.plot(kind='scatter', x='year', y='total', figsize=(10, 6)

plt.title('Australia Population 1970 - 2022')
plt.xlabel('Year')
plt.ylabel('Population')

# plot line of best fit
plt.plot(x_au, fit_au[0] * x_au + fit_au[1], color='red') # recall
plt.annotate('y={0:.0f} x + {1:.0f}'.format(fit_au[0], fit_au[1]),

plt.show()

# print out the line of best fit
'Australia Population = {0:.0f} * Year + {1:.0f}'.format(fit_au[0],
```



Out[91]: 'Australia Population = 262563 * Year + -505224546'

Comparing China and India Population

In [92]:
```
# let's compare China and India population

# creating dataframe for both countries
df_CI = df_copy.loc[['China', 'India'], years]
df_CI
```

Out[92]:

Country	1970	1980	1990	2000	2010	2015	2
China	822534450	982372466	1153704252	1264099069	1348191368	1393715448	1424929
India	557501301	696828385	870452165	1059633675	1240613620	1322866505	139638

In [93]:
```
# let's transpose the dataframe 'df_CI'

df_CI = df_CI.transpose()
df_CI.head()
```

Out[93]:

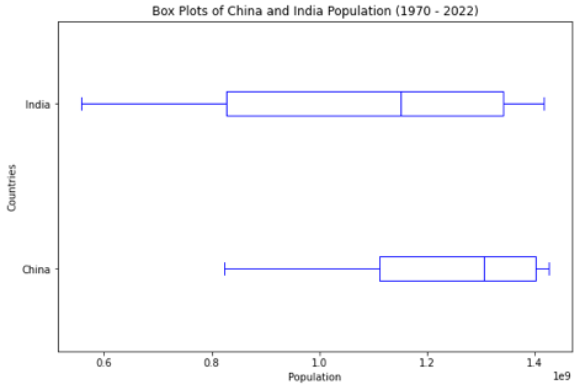
Country	China	India
1970	822534450	557501301
1980	982372466	696828385
1990	1153704252	870452165
2000	1264099069	1059633675
2010	1348191368	1240613620

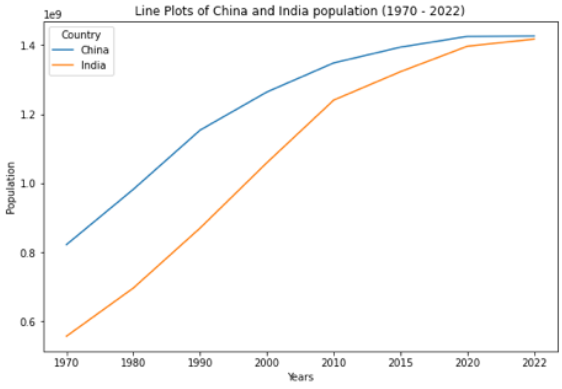
In [94]:
```
# checking out the statistical summary quickly

df_CI.describe()
```

Out[94]:

Country	China	India
count	8.000000e+00	8.000000e+00
mean	1.226929e+09	1.070182e+09
std	2.240527e+08	3.299235e+08
min	8.225344e+08	5.575013e+08
25%	1.110871e+09	8.270462e+08
50%	1.306145e+09	1.150124e+09
75%	1.401519e+09	1.341247e+09
max	1.425887e+09	1.417173e+09

In [95]:
```
# let's perform a side by side comparison of the box plot with the

fig = plt.figure()

ax0 = fig.add_subplot(1, 2, 1)
ax1 = fig.add_subplot(1, 2, 2)

# Subplot 1: Box plot
df_CI.plot(kind='box', color='blue', vert=False, figsize=(20, 6), a
ax0.set_title('Box Plots of China and India Population (1970 - 2022
ax0.set_xlabel('Population')
ax0.set_ylabel('Countries')

# Subplot 2: Line plot
df_CI.plot(kind='line', figsize=(20, 6), ax=ax1)
ax1.set_title ('Line Plots of China and India population (1970 - 20
ax1.set_ylabel('Population')
ax1.set_xlabel('Years')

plt.show()
```





Data Normalization (China - India)

In [96]:
```
df_norm = df_copy[years].transpose() # transposed dataframe

df_norm.index = map(int, df_norm.index) # cast the Years (the index

# let's label the index. This will automatically be the column name
df_norm.index.name = 'Year'

df_norm.reset_index(inplace=True) # reset index to bring the Year i

df_norm.head(3)
```

Out[96]:

Country	Year	China	India	United States	Indonesia	Pakistan	Nigeria	B
0	1970	822534450	557501301	200328340	115228394	59290872	55569264	9636
1	1980	982372466	696828385	223140018	148177096	80624057	72951439	12228
2	1990	1153704252	870452165	248083732	182159874	115414069	95214257	15070

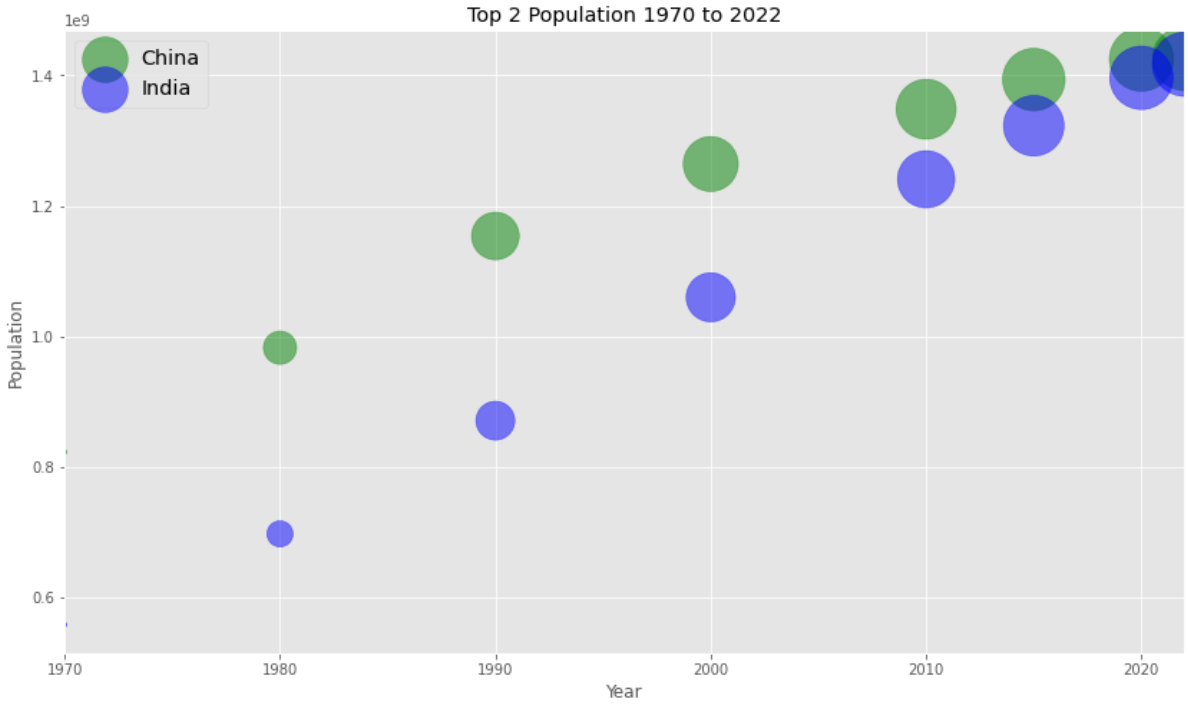
3 rows × 235 columns

In [97]:
```
# normalize China data
norm_china = (df_norm['China'] - df_norm['China'].min()) / (df_norm

# normalize India data
norm_india = (df_norm['India'] - df_norm['India'].min()) / (df_norm
```

In [98]:
```

```

```
# plotting the graph for normalized China - India

import matplotlib.pyplot as plt
%matplotlib inline
import matplotlib as mpl
mpl.style.use(['ggplot'])

# China
ax0 = df_norm.plot(kind='scatter',
                   x='Year',
                   y='China',
                   figsize=(14, 8),
                   alpha=0.5,  # transparency
                   color='green',
                   s=norm_china * 2000 + 10,  # pass in weights
                   xlim=(1970, 2022)
                   )

# India
ax1 = df_norm.plot(kind='scatter',
                   x='Year',
                   y='India',
                   alpha=0.5,
                   color="blue",
                   s=norm_india * 2000 + 10,
                   ax=ax0
                   )

ax0.set_ylabel('Population')
ax0.set_title('Top 2 Population 1970 to 2022')
ax0.legend(['China', 'India'], loc='upper left', fontsize='x-large')
```

Out[98]: <matplotlib.legend.Legend at 0x7f5eb6893d90>



In [ ]: